TutorialsPoint 移动端教程

wizardforcel



Published with GitBook

目錄

介紹 0
Android开发教程 1
Android 开发环境配置 - Android开发教程 1.1
Android 架构 - Android开发教程 1.2
Android 应用组件 - Android开发教程 1.3
Android Hello World示例 - Android开发教程 1.4
Android 资源组织和访问 - Android开发教程 1.5
Android Activity - Android开发教程 1.6
Android Service - Android开发教程 1.7
Android广播接收器 - Android开发教程 1.8
Android内容提供者 - Android开发教程 1.9
Android碎片/片段 - Android开发教程 1.10
Android Intent过滤器 - Android开发教程 1.11
Android UI布局 - Android开发教程 1.12
Android UI控件 - Android开发教程 1.13
Android事件处理 - Android开发教程 1.14
Android样式和主题 - Android开发教程 1.15
Android自定义组件 - Android开发教程 1.16
Android拖放 - Android开发教程 1.17
Android通知 - Android开发教程 1.18
Android基于位置服务 - Android开发教程 1.19
Android发送电子邮件 - Android开发教程 1.20
Android发送短信/SMS - Android开发教程 1.21
Android拨打电话 - Android开发教程 1.22
发布Android应用 - Android开发教程 1.23
ionic 教程 2
ionic 入门 2.1
ionic 简介 2.1.1
ionic 安装 2.1.2
ionic 创建 APP 2.1.3
ionic CSS 2.2
ionic 头部与底部 2.2.1
ionic 按钮 2.2.2
ionic 列表 2.2.3
ionic 卡片 2.2.4

ionic 表单和输入框 2.2.5
ionic Toggle(切换开关) 2.2.6
ionic 单选框 2.2.7
ionic Range 2.2.8
ionic select 2.2.9
ionic tab(选项卡) 2.2.10
ionic 网格(Grid) 2.2.11
ionic 颜色 2.2.12
ionic icon(图标) 2.2.13
ionic JavaScript 2.3
ionic 上拉菜单(ActionSheet) 2.3.1
ionic 背景层 2.3.2
ionic 下拉刷新 2.3.3
ionic 复选框 2.3.4
ionic 单选框操作 2.3.5
ionic 切换开关操作 2.3.6
ionic 手势事件 2.3.7
ionic 头部和底部 2.3.8
ionic 列表操作 2.3.9
ionic 加载动作 2.3.10
ionic 模型 2.3.11
ionic 导航 2.3.12
ionic 平台 2.3.13
ionic 浮动框 2.3.14
ionic 对话框 2.3.15
ionic 滚动条 2.3.16
ionic 侧栏菜单 2.3.17
ionic 滑动框 2.3.18
ionic 加载动画 2.3.19
ionic 选项卡栏操作 2.3.20
IOS 教程 3
IOS 简介 3.1
Objective-C 简介 3.2
创建第一款iPhone应用程序 3.3
IOS通用应用程序 3.4
IOS相机管理 3.5
IOS定位操作 3.6
IOS SQLite数据库 3.7
IOS发送电子邮件 3.8

IOS音频和视频(Audio & Video) 3.9
IOS文件处理 3.10
IOS地图开发 3.11
IOS应用内购买 3.12
IOS iAD整合 3.13
IOS GameKit 3.14
IOS 故事板(Storyboards) 3.15
IOS自动布局 3.16
IOS-Twitter和Facebook 3.17
IOS内存管理 3.18
IOS应用程序调试 3.19
jQuery Mobile 教程 4
jQuery Mobile 简介 4.1
jQuery Mobile 安装 4.2
jQuery Mobile 页面 4.3
jQuery Mobile 页面切换 4.4
jQuery Mobile 按钮 4.5
jQuery Mobile 按钮图标 4.6
jQuery Mobile 工具栏 4.7
jQuery Mobile 导航栏 4.8
jQuery Mobile 可折叠块 4.9
jQuery Mobile 网格 4.10
jQuery Mobile 列表视图 4.11
jQuery Mobile 列表内容 4.12
jQuery Mobile 表单 4.13
jQuery Mobile 表单输入元素 4.14
jQuery Mobile 表单选择菜单 4.15
jQuery Mobile 表单滑动条 4.16
jQuery Mobile 主题 4.17
jQuery Mobile 事件 4.18
jQuery Mobile 触摸事件 4.19
jQuery Mobile 滚屏事件 4.20
jQuery Mobile 方向改变事件 4.21
jQuery Mobile Data 属性 4.22
jQuery Mobile 图标 4.23
jQuery Mobile 事件 4.24
jQuery Mobile 页面事件 4.25
jQuery Mobile CSS 类 4.26

TutorialsPoint 移动端教程

Android开发教程

Android是一个开源的，基于Linux 的移动设备操作系统，如智能手机和平板电脑。Android 是由开放手机联盟和谷歌的带领下与其他公司开发的。

Android 提供了一个统一的应用程序开发方法，这意味着开发人员只需要开发Android，并且他们的应用程序应该能够运行在不同搭载Android移动设备。

谷歌在2007年发布第一个商业版本的Android1.0，发布于2008年9月发布了第一个测试版本的Android软件开发工具包（SDK）。

2012年6月27日，在谷歌I / O大会上，谷歌宣布发布了Android版本4.1 Jelly Bean。 Jelly Bean是一个渐进的更新，改进用户界面为主要目的，无论是在功能和性能方面。

Android 源代码是根据自由和开放源码软件许可证。谷歌发布的大部分代码根据Apache许可证2.0版，Linux内核的变化根据GNU通用公共许可证版本2。

Android的特点

Android 是一个功能强大的操作系统与苹果4GS竞争，并支持强大的功能。少数几种功能列举如下：

特点	描述
漂亮的UI	Android操作系统的基本屏幕提供了一个美丽而直观的用户界面。
连通性	GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC 和 WiMAX.
存储	使用SQLite轻量的关系数据库，用于数据存储目的。
媒体支持	H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, 和 BMP
短信/消息	SMS 和 MMS
Web浏览器	基于开源的WebKit布局引擎，再加上支持HTML5和CSS3 Chrome的V8 JavaScript引擎。
多点触控	Android已经多点触控，它最初获得手机提供原生支持，如 HTC Hero。
Multi-tasking	用户可以跳从一个任务到另一个任务，并且相同时间可以同时运行各种应用。
Resizable widgets	Widgets是可调整大小，这样用户就可以扩大更多的内容或缩小以节省空间
Multi-Language	支持单向和双向文本。
GCM	谷歌云消息（GCM）是一种服务，让开发人员发送短消息数据，对Android设备的用户，而无需专有的同步解决方案。
Wi-Fi Direct	一种技术，可以让应用程序发现和对直接通过高带宽的对等网络连接。
Android Beam	一个流行的基于NFC的技术，使用户能够即时共享，只需通过触摸NFC功能将两个手机连在一起。

Android 应用程序

通常在Java语言中使用Android软件开发工具包开发Android 应用程序。

系统开发出来以兵力，Android应用程序可以轻松地打包和销售商店，可以通过如谷歌播放或亚马逊Appstore。

Android 在世界各地190多个国家数以百万计的移动设备。这是任何移动平台和快速增长的最大的安装基础。全球每天有超过100万个新的Android设备被激活。

本教程目的是教你如何开发并将Android 应用程序打包。我们将从Android应用程序编程环境设置开始，然后是Android 各个方面的应用程序。

Android 开发环境配置 - Android开发教程

Android 应用程序开发可以用以下操作系统开发：

- Microsoft Windows XP或更高版本。
- Mac OS X10.5.8或更高版本（英特尔芯片）。
- 包括GNU C库2.7或更高版本的Linux系统。

第二是开发Android应用程序所需的所有工具都是免费的，可以从网上下载。以下是软件的列表，开始 Android 应用程序编程需要用到。

- Java JDK5 或 JDK6
- Android SDK
- Eclipse IDE for Java Developers (optional)
- Android Development Tools (ADT) Eclipse Plugin (optional)

这里最后两个组件是可选的，如果正在使用Windows这些组件容易获得，都是基于Java应用程序开发。因此，让我们来看看如何进行设置所需的环境。 注：如果你觉得以下步骤比较复杂，那么可以直接使用 android 开发软件集成包：adt-bundle-windows （自行搜索）

第1步 - 安装Java开发工具包（JDK）

可以下载最新版本的Java JDK，从Oracle的Java网站：Java SE下载。安装下载的JDK文件，按照给定的说明来安装和配置设置。最后设置PATH和JAVA_HOME环境变量来引用该目录包含javac和java，通常分别为 java_install_dir/bin 和 java_install_dir。

如果运行的是Windows，把JDK安装在C:jdk1.6.0_15，在 C:autoexec.bat 文件添加以下内容：

```
set PATH=C:jdk1.6.0_15in;%PATH%
set JAVA_HOME=C:jdk1.6.0_15
```

另外，也可以右键单击“我的电脑”，选择“属性”=》”高级“=》“环境变量”，按下“确定”按钮，然后会更新 PATH 值。

在Linux上，如果SDK安装在 /usr/local/jdk1.6.0_15，如果使用的是C shell，把下面的代码到 .cshrc文件。

```
setenv PATH /usr/local/jdk1.6.0_15/bin:$PATH
setenv JAVA_HOME /usr/local/jdk1.6.0_15
```

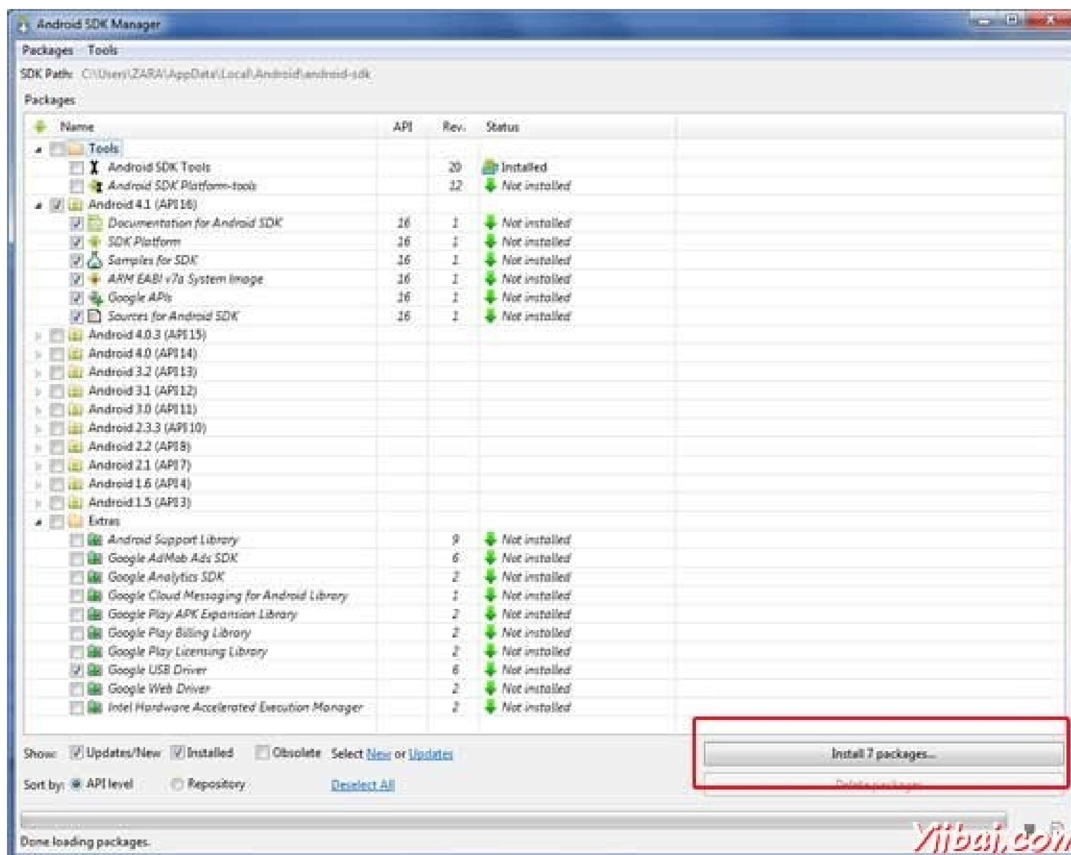
另外，如果使用Eclipse 集成开发环境（IDE），那么它会自动知道已安装Java。

第2步 - 安装Android SDK

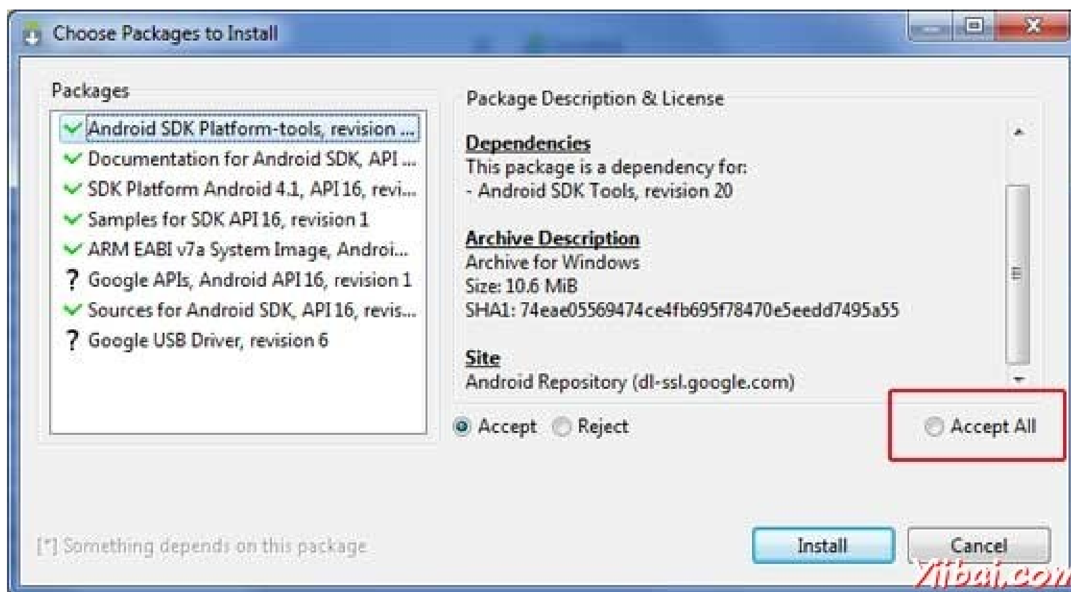
可以下载最新版本的Android SDK，从Android 官方网站：Android SDK下载。如果Windows机器上安装SDK，那么会发现一个installer_rXX windows.exe文件，只需下载并运行这个EXE将推出Android SDK工具安装向导来指导整个的安装，所以只要按照指示安装。最后，必须在机器上安装Android SDK 工具。

本教程使用的是 Windows 7操作系统环境。

因此，让我们打开 Android SDK 管理器中的选项，所有程序“>”Android SDK 工具> SDK管理器，这会给定下面的窗口：



打开 SDK管理器后，它需要点时间来安装其他所需的软件包。默认情况下，它会列出共7个要安装的软件包，但建议选择为 Android SDK 文档和SDK包的样例代码，以减少安装时间。下一步点击安装包按钮继续进行，这将显示以下对话框：



如果同意安装所有软件包，请选择接受所有的单选按钮，并继续进行通过单击“Installbutton”。现在，让SDK管理器自己工作，等待直到所有的包都安装。这可能需要一些时间，这取决于您的互联网连接。所有的包都安装完成后，SDK管理器可以关闭，使用右上方的交叉按钮。

第3步 - 安装Eclipse IDE

所有的例子已经写在本教程中使用的是Eclipse IDE。所以建议机器上安装最新版本的Eclipse。

安装Eclipse IDE，http://www.eclipse.org/downloads/下载最新的Eclipse二进制文件。下载安装：解压二进制分发到一个方便的位置。例如，在 C:eclipse，或 /usr/local/eclipse 在 Linux并适当设置PATH 变量。

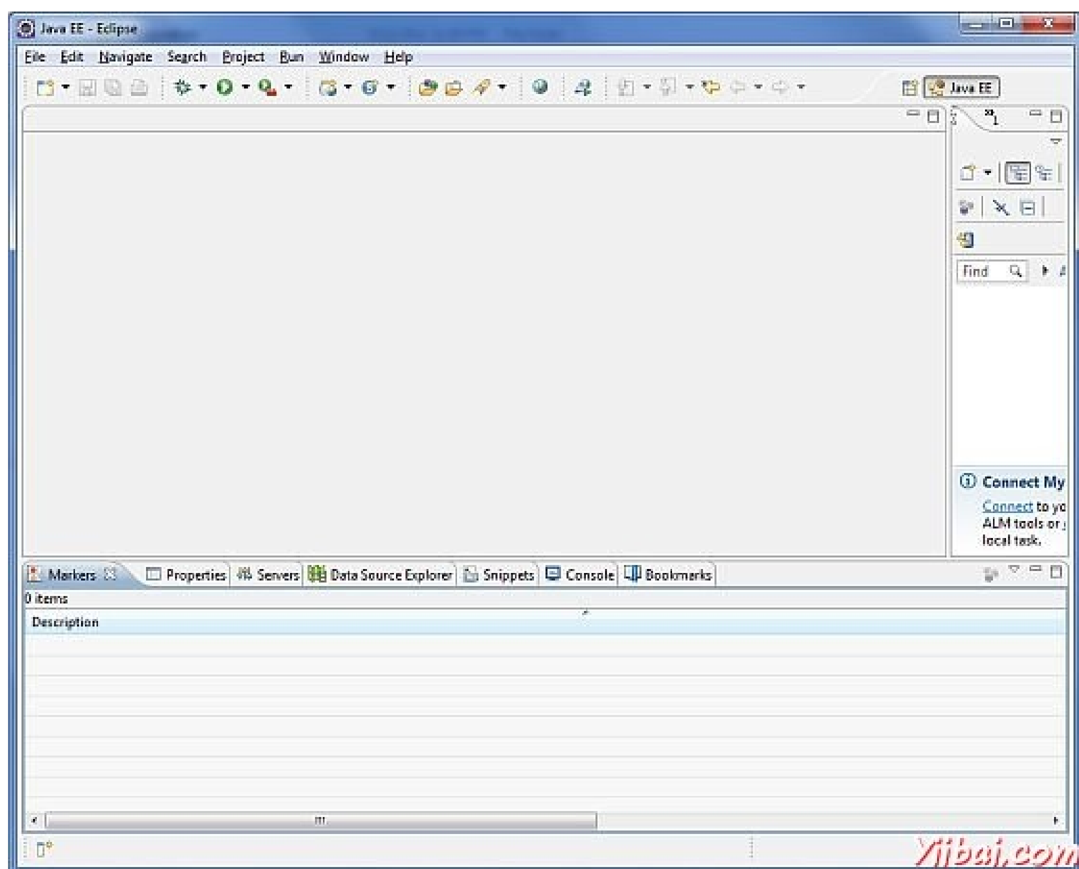
Eclipse可以启动Windows机器上执行以下命令，或者可以简单地双击 eclipse.exe

```
%C:eclipseeclipse.exe
```

Eclipse可以启动Linux机器上执行以下命令：

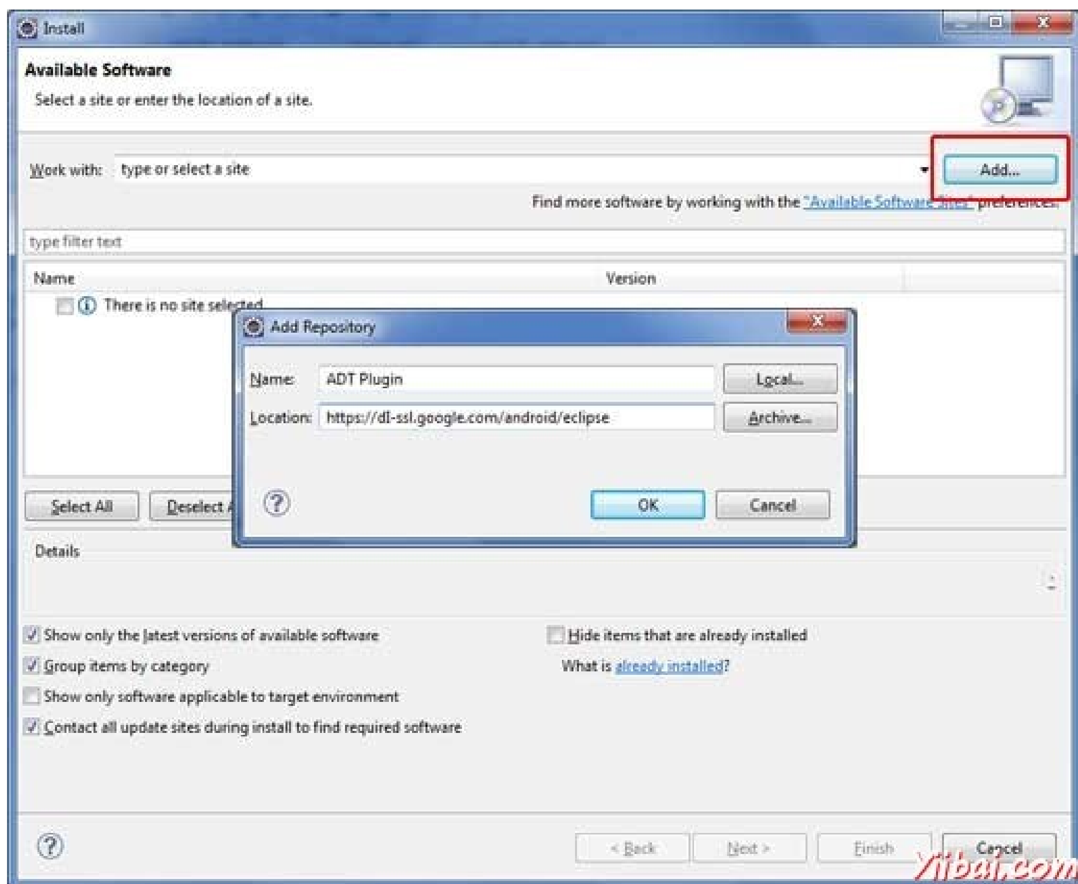
```
$/usr/local/eclipse/eclipse
```

成功启动后，如果一切正常，那么它应显示以下结果：

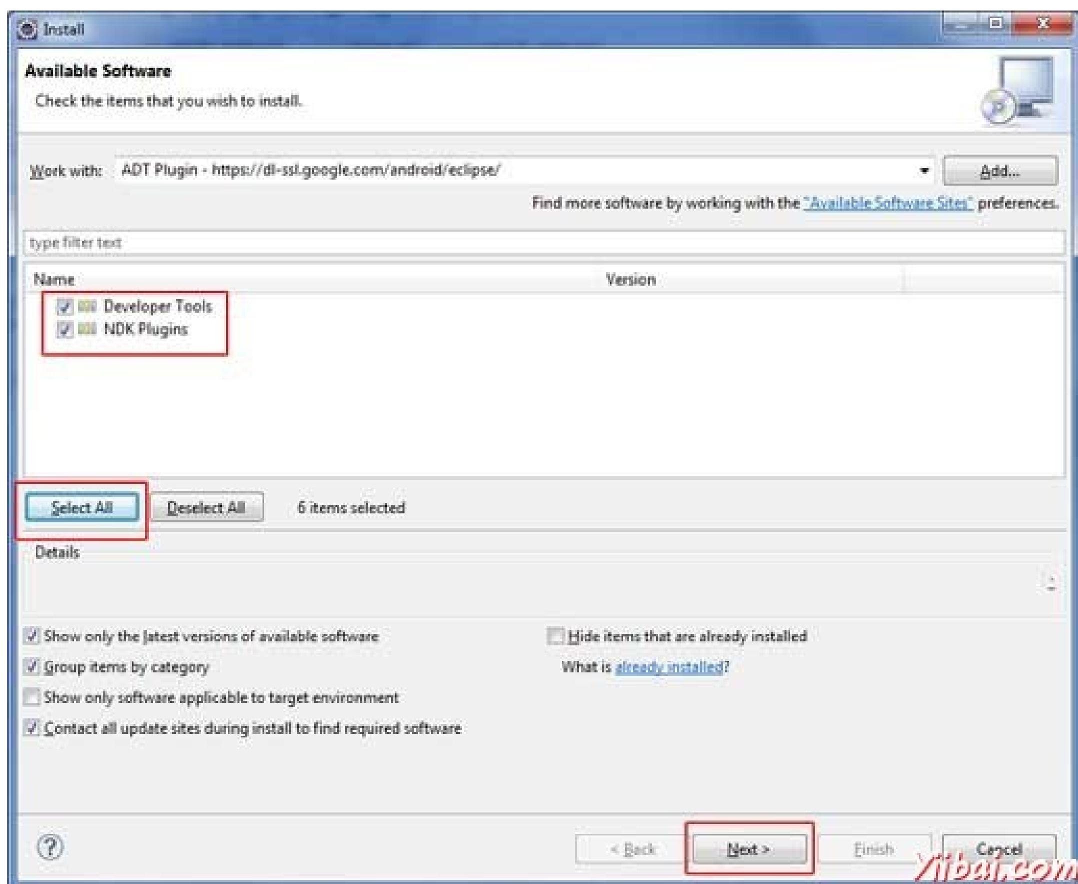


第4步 - 安装Android开发工具（ADT）插件

这个步骤将设置 Android 开发工具Eclipse插件。让我们开始打开 Eclipse，然后选择 **Help > Software Updates > Install New Software"**。这将显示以下对话。



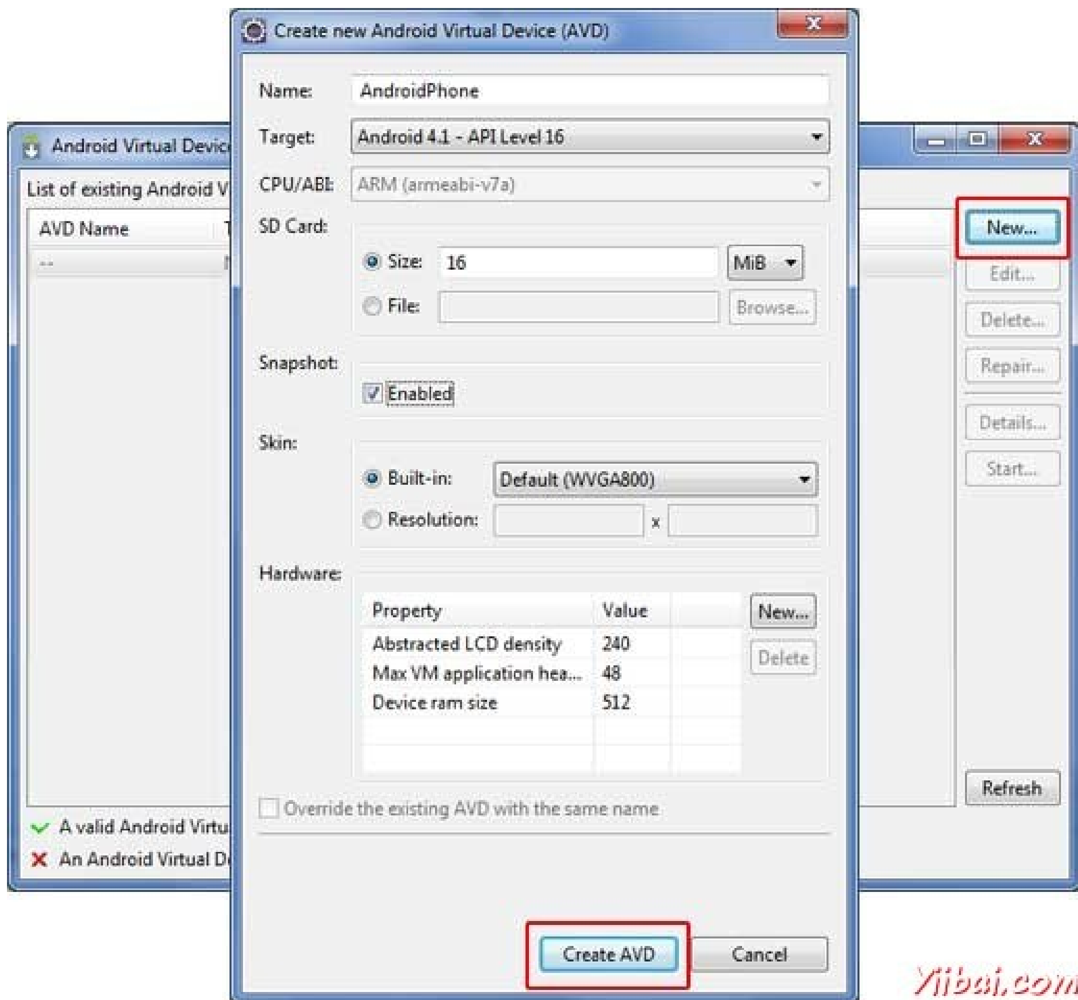
现在，使用"**Add** "按钮来添加名称和ADT插件的位置 https://dl-ssl.google.com/android/eclipse/ 。然后单击“确定”添加这个位置，只要单击“OK ”按钮，添加这个位置时，Eclipse开始搜索插件在给定的位置，最后列出找到的插件。



现在，选择列出的所有插件使用"**Select All**"按钮，并单击“**Next** ”按钮，将引导提前安装Android开发工具和其他所需的插件。

第5步 - 创建Android虚拟设备

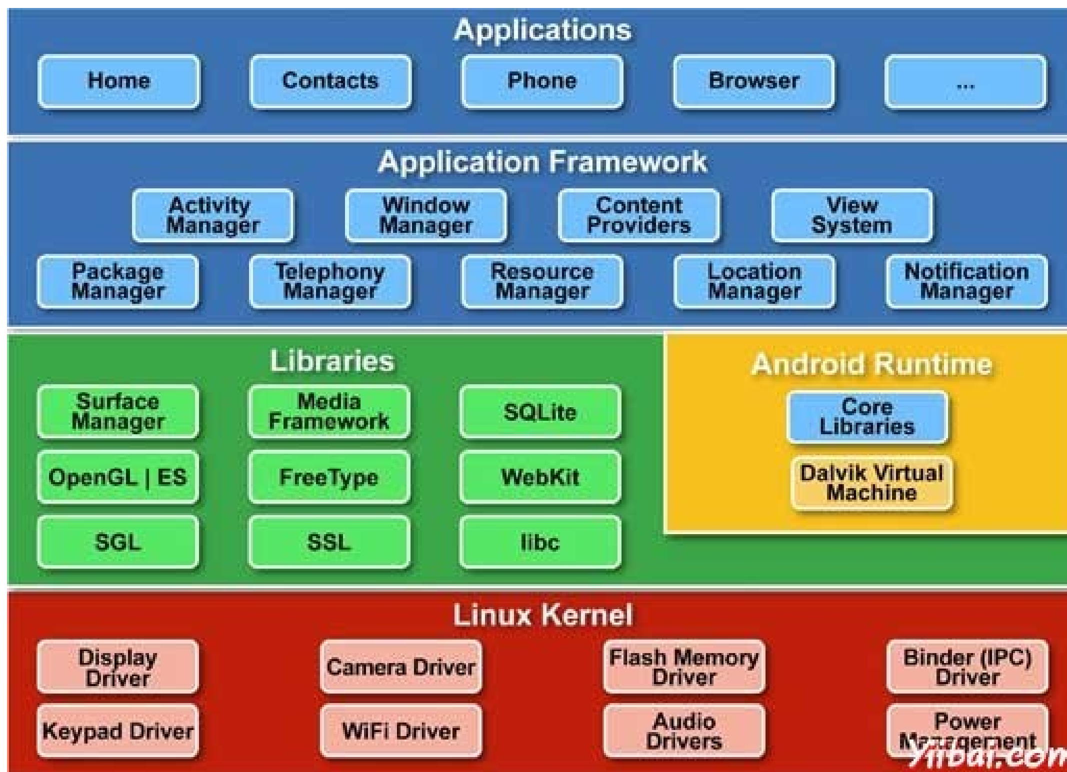
为了测试Android应用程序，需要一个虚拟的 Android设备。所以，在我们开始之前，在代码创建一个Android 虚拟设备。打开 Android AVD管理使用Eclipse菜单选项 **Window > AVD Manager>**将启动 Android AVD管理。使用“**New**”按钮，创建一个新的Android虚拟设备，并输入以下信息，然后点击"**Create AVD**"按钮。



如果AVD创建成功，这意味着Android应用程序开发环境已经配置完成。使用右上角的十字按钮关闭此窗口。最好重新启动计算机，一旦完成了这最后一步，就可以准备进行第一个 Android例子了，但在此之前，我们将看到Android应用开发相关的一些比较重要的概念。

Android 架构 - Android开发教程

Android操作系统是一个堆栈的软件组件，它大致分为五个部分和四个主要层的体系结构，如下图所示。



Linux内核

在层次的底部是 Linux - Linux 2.6。它提供基本的系统功能，如进程管理，内存管理，设备管理，如：相机，键盘，显示器等内核处理的事情，Linux 确实不错，如网络设备驱动程序比较多，内搭外围硬件接口。

程序库

在Linux内核之上，有一个组库，包括开放源码的 Web浏览器引擎WebKit，libc库，SQLite数据库，这是一个非常有用的库，用于存储和共享应用程序数据，播放和录制音频和视频，SSL库负责互联网安全等。

Android运行时

这是体系结构第三个部分并在第二层之上由底部提供。本节提供了一个关键的组件，叫做Java虚拟机，是一种专门设计和优化的 Android Dalvik 虚拟机。

Dalvik虚拟机使用的Linux核心功能，如内存管理和多线程，在Java语言中是内在的。 Dalvik虚拟机将每一个Android应用程序运行在自己的进程中，使用Dalvik虚拟机实例。

Android还提供了一组核心库，使Android应用程序开发人员使用标准的Java编程语言编写Android应用程序。

应用程序框架

应用程序框架层使用Java类形式的应用程序提供了许多的更高级别的服务。允许应用程序开发人员在其应用程序中使用这些服务。

应用

在最上层，即所有的 Android 应用程序。一般我们编写的应用程序只被安装在这层。应用的例子如：浏览器，游戏等。

Android 应用组件 - Android开发教程

应用组件是一个Android应用程序的基本构建块。这些组件是松耦合的应用程序清单文件AndroidManifest.xml中介绍了每种组件的应用程序，以及它们如何相互作用。

有以下四个主要组成部分，可用在一个Android应用程序：

组件	描述
Activities	他们决定了用户界面和处理用户交互，智能手机的屏幕
Services	他们处理与应用程序相关的后台处理。
Broadcast Receivers	他们处理的Android操作系统和应用程序之间的通信。
Content Providers	他们处理的数据和数据库管理方面的问题。

Activities

一个活动（activity）表示一个单一的屏幕上的用户界面。例如，电子邮件应用程序可能有一个活动，显示新的电子邮件列表，另一个活动，撰写电子邮件，阅读电子邮件和其他活动。如果应用程序有一个以上的活动，然后其中一人应标记为活动启动应用程序时提出。

被实现为一个活动Activity类的子类，如下：

```
public class MainActivity extends Activity {

}
```

Services

服务是一种在后台运行的组件来执行长时间运行的操作。例如，一个服务可以在后台播放音乐，而用户在不同的应用程序，或者它可能通过网络获取数据，而不阻塞用户交互与活动。

实现一个service作为一个服务类的子类如下：

```
public class MyService extends Service {

}
```

广播接收器

广播接收机简单地响应从其他应用程序或从系统广播消息。例如，应用程序也可以发起广播，以让其他应用程序知道某些数据已经被下载到设备上，可供他们使用，所以这是广播接收器，会拦截此通信，并会采取适当行动。

广播接收机被实现为BroadcastReceiver的类的子类，每个消息被作为一个Intent对象广播。

```
public class MyReceiver  extends  BroadcastReceiver {

}
```

内容提供者

内容提供者组件提供数据从一个应用到其他要求。 ContentResolver类的方法，通过这样的请求的处理。这些数据可以被存储在文件系统中，数据库或其他地方。

内容提供商实现 ContentProvider类的子类，必须实施了一套标准的API，使其他应用程序来执行交易的。

```
public class MyContentProvider extends  ContentProvider {

}
```

我们将通过这些标签涵盖应用程序组件的细节，同时在单独的章节。

附加组件

附加组件可以使用在上述的实体，它们的逻辑以及它们之间的连线构造。这些组件包括：

组件	描述
Fragments	表示的行为或在活动中的用户界面的一部分。
Views	绘制的屏幕上的按钮的UI元素，列表形式等。
Layouts	查看层次结构，控制屏幕格式和外观视图。
Intents	消息连线组件在一起。
Resources	外部因素，如字符串，常数和可绘制的图片。
Manifest	应用程序的配置文件。

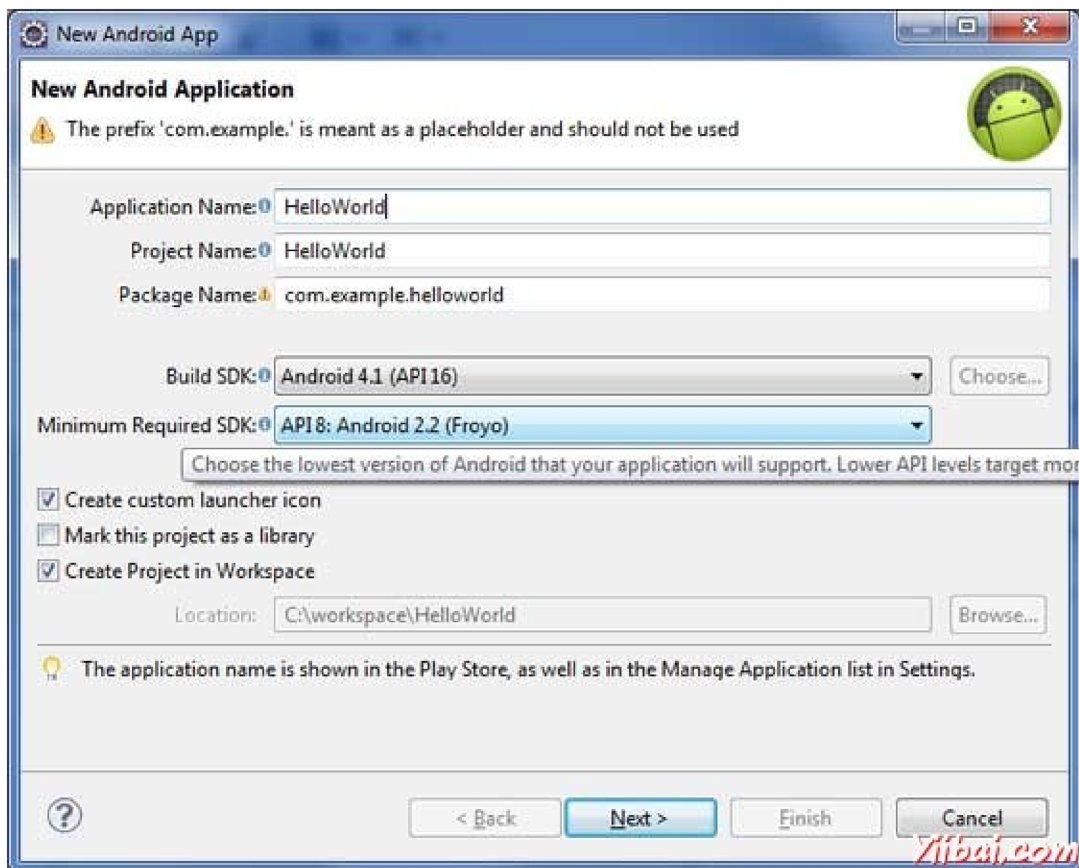
Android Hello World示例 - Android开发教程

让我们开始实际的Android编程。在开始使用Android SDK编写的第一个例子之前，必须确保设置正确的Android开发环境 - 环境设置教程。也假设你有一点 Eclipse IDE 操作知识。

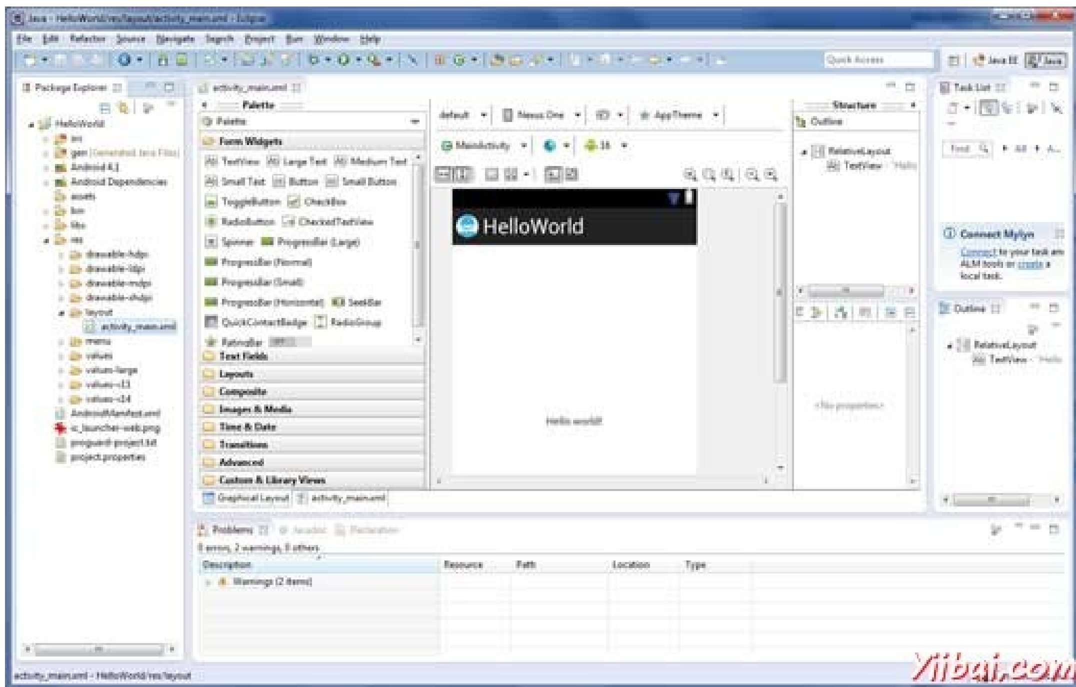
因此，现在编写一个简单的Android应用程序，它将打印 "Hello World!".

创建Android应用

第一步：使用Eclipse IDE创建一个简单的Android应用程序。按照选择File -> New -> Project， 最后从向导列表选择 Android New Application 。现在，应用程序命名为 HelloWorld 使用向导窗口如下：

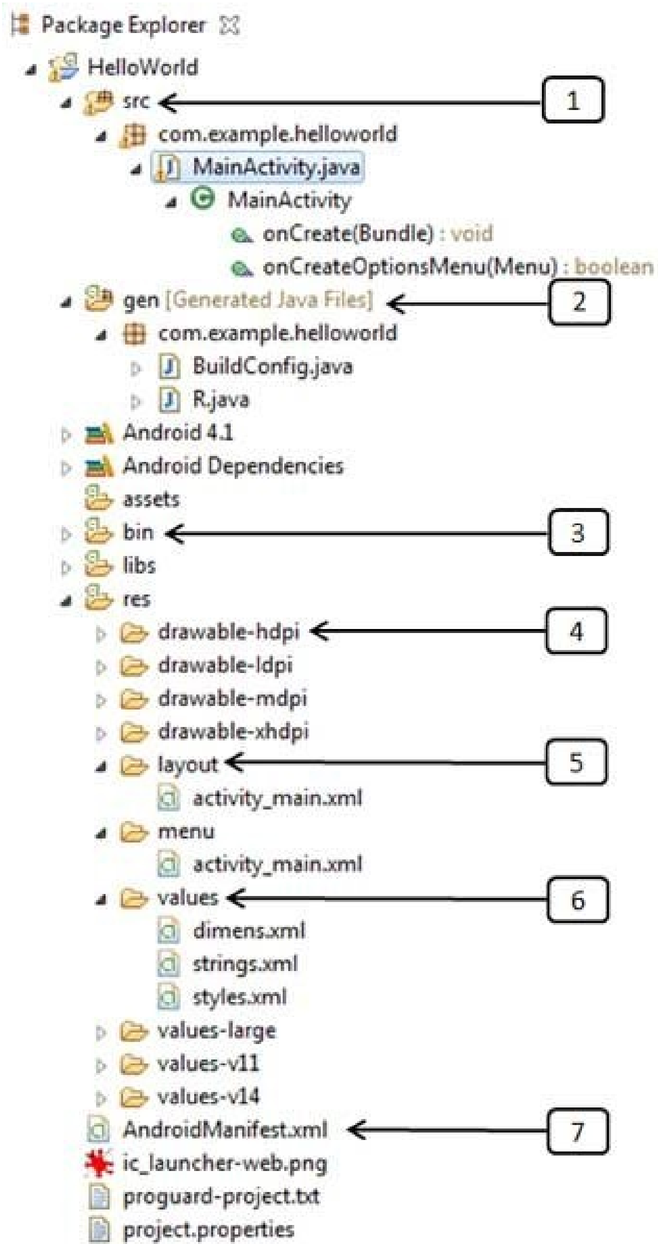


接下来，按照所提供的指示，并保持所有其他项为默认，直到最后一步。项目创建成功后，将有以下项目的画面：



Android应用解剖

在运行应用程序之前，应该注意在Android项目的几个目录和文件：


Package Explorer
HelloWorld
src
com.example.helloworld
MainActivity.java
MainActivity
onCreate(Bundle) : void
onCreateOptionsMenu(Menu) : boolean
gen [Generated Java Files]
com.example.helloworld
BuildConfig.java
R.java
Android 4.1
Android Dependencies
assets
bin
libs
res
drawable-hdpi
drawable-ldpi
drawable-mdpi
drawable-xhdpi
layout
activity_main.xml
menu
activity_main.xml
values
dimens.xml
strings.xml
styles.xml
values-large
values-v11
values-v14
AndroidManifest.xml
ic_launcher-web.png
proguard-project.txt
project.properties
1
2
3
4
5
6
7
Yiibai.com

S.N.	文件夹、文件和说明
1	**src** 这包含项目java源文件。默认情况下，它包括一个 MainActivity.java 源文件的活动类，应用程序运行时使用的应用程序图标启动
2	**gen** 这包含了R文件，编译器生成的文件在项目中找到的所有资源。不要修改这个文件。
3	**bin** 此文件夹包含了Android APK包文件。ADT在创建生成过程中一切需要运行一个Android应用程序
4	**res/drawable-hdpi** 被设计用于高密度的屏幕可绘制的对象目录
5	**res/layout** 这是一个目录文件，定义应用程序的用户界面
6	**res/values** 这是一个目录，包含各种XML文件，其中包含资源的集合，如字符串和颜色定义
7	**AndroidManifest.xml** 这是文件列表，该文件描述了应用程序的基本特征，并定义每个组件

下面的部分将简要介绍一些重要的应用程序文件。

主要活动Activity 文件

主要活动代码是Java文件：MainActivity.java。这是实际的应用程序文件，最终被转换 Dalvik 可执行的文件，并运行应用程序。以下是默认的Hello World 应用的程序向导生成的代码：

```
package com.example.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.MenuItem;
import android.support.v4.app.NavUtils;

public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
}
```

在这里，R.layout.activity_main 是指位于在 res/layout 文件夹的 activity_main.xml 文件。onCreate()方法是加载一个活动时，被触发的许多方法之一。

Manifest 文件

组件是开发应用程序的一部分，必须声明其所有组件在 AndroidManifest.xml 中，应用程序项目的根目录在 amanifest 文件。此文件作为Android操作系统和应用程序之间的接口。例如，默认 manifest 文件将看起来如以下文件：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/androic
   package="com.example.helloworld"
   android:versionCode="1"
   android:versionName="1.0" >
   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="15" />
   <application
       android:icon="@drawable/ic_launcher"
       android:label="@string/app_name"
       android:theme="@style/AppTheme" >
       <activity
           android:name=".MainActivity"
           android:label="@string/title_activity_main" >
           <intent-filter>
               <action android:name="android.intent.action.MAIN" /:
               <category android:name="android.intent.category.LAUN
           </intent-filter>
       </activity>
   </application>
</manifest>
```

这里<application>...</application>标记括起相关应用程序的组件。属性android:icon将指向应用程序图标，可在res/drawable-hdpi 之下找到。应用程序使用位于图像绘制文件夹，文件名为 ic_launcher.png 。

<activity>标签是用来指定活动，属性android:name指定活动子类，android类完全限定android:label 属性指定一个字符串作为标签使用的活动。可以使用<activity>标签指定多个活动。

intent 过滤器的动作被命名为android.intent.action.MAIN，表示这个活动作为应用程序的入口点。类的意图过滤器是namedandroid.intent.category.LAUNCHER声明，应用程序可以从设备的启动图标启动。

@string是指 strings.xml文件，解释如下。@string/app_name是strings.xml文件中的app_name是“HelloWorld” 定义的字符串。类似的方式，其他的字符串填充到应用中。

以下是使用manifest 文件中指定不同的 Android 应用程序组件的标签列表：

- <activity> 活动的元素
- <service> 服务的元素
- <receiver> 广播接收器的元素
- <provider> 内容提供者的元素

Strings 文件

strings.xml文件在 res/values 文件夹，它包含了所有应用程序所使用的文本。例如，按钮，标签，默认的文本和字符串类型相似的名称都在这个文件。这个文件是负责为他们的文字内容。例如，默认字符串文件看起来像以下文件：

```
<resources>
    <string name="app_name">HelloWorld</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>
</resources>
```

R.java 文件

gen/com.example.helloworld/R.java文件就像MainActivity.java 和Java文件strings.xml等资源之间的胶水。这是一个自动生成的文件，不要修改R.java文件的内容。R.java文件下面是一个示例：

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package com.example.helloworld;

public final class R {
    public static final class attr {
    }
    public static final class dimen {
        public static final int padding_large=0x7f040002;
        public static final int padding_medium=0x7f040001;
        public static final int padding_small=0x7f040000;
    }
    public static final class drawable {
        public static final int ic_action_search=0x7f020000;
        public static final int ic_launcher=0x7f020001;
    }
    public static final class id {
        public static final int menu_settings=0x7f080000;
    }
    public static final class layout {
        public static final int activity_main=0x7f030000;
    }
    public static final class menu {
        public static final int activity_main=0x7f070000;
    }
    public static final class string {
        public static final int app_name=0x7f050000;
        public static final int hello_world=0x7f050001;
        public static final int menu_settings=0x7f050002;
        public static final int title_activity_main=0x7f050003;
    }
    public static final class style {
        public static final int AppTheme=0x7f060000;
    }
}
```

布局文件

activity_main.xml是在res/layout 目录，所引用应用程序接口布局文件。要修改应用程序的布局这个文件非常频繁地被修改。“Hello World ”应用中，这个文件默认布局，内容如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent" >

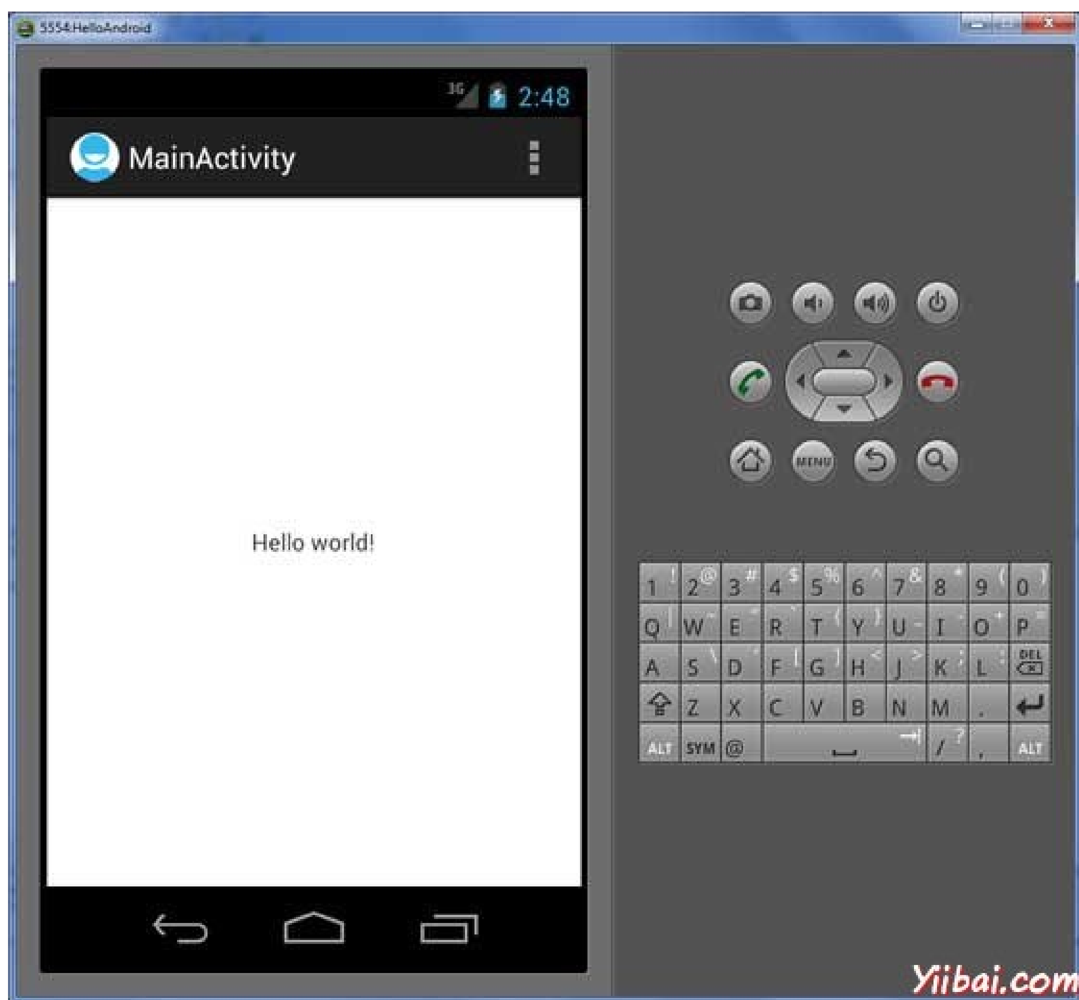
   <TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_centerHorizontal="true"
       android:layout_centerVertical="true"
       android:padding="@dimen/padding_medium"
       android:text="@string/hello_world"
       tools:context=".MainActivity" />

</RelativeLayout>
```

这是一个简单的相对布局，我们将在一个单独的章节学习其它复杂例子。TextView是被用来构建图形用户界面并由Android控制，它有各种各样的Android属性，如：android:layout_width, android:layout_height 等被用来设置其宽度和高度等等，@string是指位于strings.xml文件在文件夹res/values中。因此，@string/hello_world 指 hello 定义的字符串在 strings.xml 文件中的 “Hello World!” 字符串。

运行应用程序

让我们试着来运行 Hello World！应用程序。假设创建了AVD，同时做了环境设置。打开Eclipse运行应用程序，打开一个项目的活动文件，并单击“Run”工具栏上的图标。 Eclipse 在 AVD上安装的应用程序后并启动它，如果一切都设置并应用没有问题，它会显示以下模拟器窗口：



恭喜！开发的第一款Android应用程序完成，现在只要把下面剩余的教程一步一步学习，一定能成为一个熟练的 Android开发者。

Android 资源组织和访问 - Android开发教程

在许多Android项目中，有很多东西要用来建立一个良好的Android应用程序。除了编码应用各种资源，如：位图，颜色，布局定义，用户界面字符串，动画，静态内容。在res/目录下，这些资源一直保持在各自子目录。

这一小节将学习如何组织应用程序资源，指定替代资源，并在应用程序访问它们。

组织资源

将每种类型的资源在一个特定项目的 res/目录的子目录。例如，这里有一个简单的项目文件层次：

```
MyProject/
    src/
        MyActivity.java
    **res/**
        drawable/
            icon.png
        layout/
            activity_main.xml
            info.xml
        values/
            strings.xml
```

res/目录中包含不同的子目录的不同资源。在这里有一个图像资源，两个布局资源和一个字符串资源文件。下表给出了详细的项目在 res/目录里面支持的资源。

目录	资源类型
anim/	定义属性的动画XML文件。它们被保存在res/anim/文件夹，并从R.anim类访问
color/	定义的颜色状态列表的XML文件。它们被保存在res/color/，并从R.color类访问
drawable/	像被编译成位图。.png, .jpg, .gif 或XML文件，状态列表，图形，动画可绘制的图像文件。它们被保存在res/drawable/，并可从R.drawable类访问
layout/	定义用户界面布局的XML文件。它们被保存在res/layout/，并可从R.layout类访问
menu/	定义应用程序菜单，如选项菜单，上下文菜单或子菜单的XML文件。它们被保存在res/menu/，并可从R.menu类访问
raw/	任意文件保存在其原始形式。您需要callResources.openRawResource()与所述资源ID，即R.raw.filename来打开这样的原始文件
values/	包含简单的值，如字符串，整数和颜色的XML文件。例如，这里有一些文件名约定资源，可以创建在该目录中： `arrays.xml` 资源数组，并可从R.array类访问 `integers.xml` 资源整数，并可从R.integerclass访问 `bools.xml` 资源布尔型，并可从R.bool类访问 `colors.xml` 为颜色值，并可从R.color类访问 `dimens.xml` 为维度值，并可从R.dimen类访问 `strings.xml` 为字符串值，并且可从R.string类访问 `styles.xml` 为样式并可从R.style类访问
xml/	可以通过调用Resources.getXML()来读取在运行时任意的XML文件。可以在这里保存各种配置文件，这些文件可在运行时使用

替代资源

应用程序提供替代资源以支持特定的设备配置。例如，包括替代绘制资源（ie.images），针对不同的语言不同的屏幕分辨率并替代字符串资源。在运行时 Android 检测当前设备的配置，并为应用程序加载适当的资源。

要指定一组资源的配置具体的替代，请遵循以下步骤：

- 创建新目录在res/ 目录下，命名形式如 <resources_name>-<config_qualifier>。这里resources_name 是在上表中提到的资源，如layout, drawable 等限定符将指定一个单独的配置，要使用这些资源。可以查看官方文档的完整列表，限定符为不同类型的资源。
- 在这个新的目录中保存相应的替代资源。资源文件必须被命名为默认的资源文件，如下面的例子所示的完全一样，但这些文件具有特定内容的替代。例如，虽然图像的文件名是相同的，但对高分辨率屏幕，其分辨率会很高。

下面是一个例子，它指定一个默认的屏幕和高分辨率屏幕的替代图像的图像。

```
MyProject/
    src/
        MyActivity.java
    **res/**
        drawable/
            icon.png
            background.png
        **drawable-hdpi/**
            icon.png
            background.png
        layout/
            activity_main.xml
            info.xml
        values/
            strings.xml
```

下面是另一个例子，指定一种默认语言为阿拉伯语并指定替代布局。

```
MyProject/
    src/
        MyActivity.java
    **res/**
        drawable/
            icon.png
            background.png
        **drawable-hdpi/**
            icon.png
            background.png
        layout/
            activity_main.xml
            info.xml
        **layout-ar/**
            main.xml
        values/
            strings.xml
```

访问资源

在应用程序开发中，需要访问定义的资源，无论是在代码还是在布局XML文件。下面的部分介绍了如何访问资源在这两个场景：

访问资源代码

当Android应用程序被编译时，会产生一个R类，其中包含在res/ 目录中的所有可用资源的资源ID。使用R类直接访问该子目录和资源名称或资源ID。

示例：

要访问 res/drawable/myimage.png，并设置一个ImageView 可使用下面的代码：

```
ImageView imageView = (ImageView) findViewById(R.id.myimageview);
imageView.setImageResource(R.drawable.myimage);
```

下面第一行代码，使用R.id.myimageview id 为myimageview定义布局文件。第二行代码使用 R.drawable.myimage 得到的图像名称myimage， 在 /res子目录下。

示例：

考虑在下一个例子 res/values/strings.xml 有以下定义：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string  name="hello">Hello, World!</string>
</resources>
```

现在，可以设置一个TextView对象msg 文字使用资源ID如下：

```
TextView msgTextView = (TextView) findViewById(R.id.msg);
msgTextView.setText(R.string.hello);
```

示例：

考虑一个布局res/layout/activity_main.xml 如以下的定义：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >
<TextView android:id="@+id/text"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Hello, I am a TextView" />
<Button android:id="@+id/button"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Hello, I am a Button" />
</LinearLayout>
```

这个应用程序代码的一个活动将加载此布局，在onCreate()方法如下：

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_activity);
}
```

XML中访问资源

考虑下面的XML资源res/values/strings.xml文件，包括颜色资源和一个字符串资源：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="opaque_red">#f00</color>
   <string name="hello">Hello!</string>
</resources>
```

现在，可以利用这些资源，在下面的布局文件中设置文本颜色和文本字符串如下：

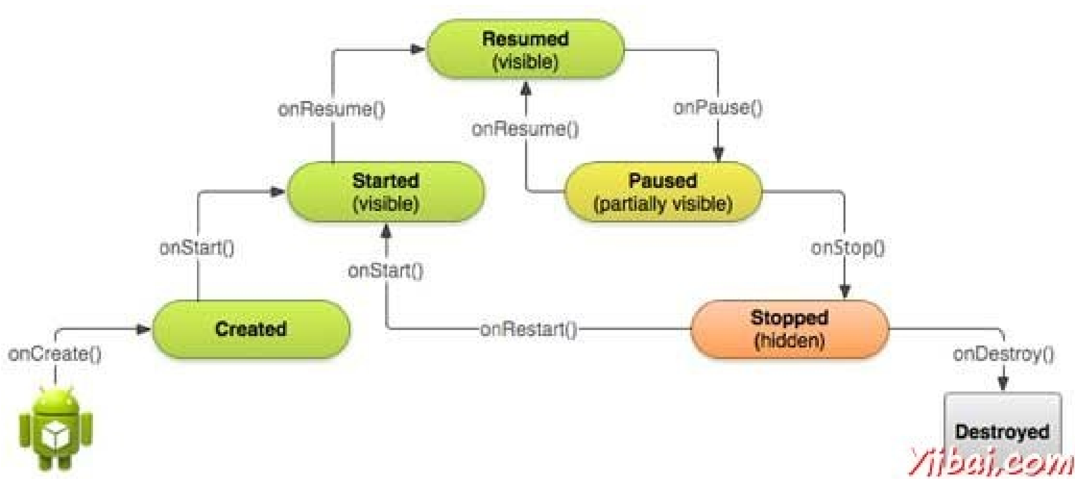
```
<?xml version="1.0" encoding="utf-8"?>
<EditText xmlns:android="http://schemas.android.com/apk/res/androic
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:textColor=**"@color/opaque_red"**
    android:text=**"@string/hello" />**
```

现在，如果将再次通过前面的章节了解，例如：Hello World！应用，将有助于更好的理解本小节介绍的概念。了解在前面的小节中是如何利用各种资源的基本操作。

Android Activity - Android开发教程

活动(activity)表示一个单一屏幕上的用户界面。例如，电子邮件应用程序可能是一个活动，显示新的电子邮件列表是另一个活动，撰写电子邮件，阅读电子邮件可能又是其它的活动。如果应用程序有一个以上的活动，那么应该将其中的一个活动标记为活动启动应用程序。

如果曾使用C，C++或Java编程语言，那么可以知道一般程序是从 main() 函数开始。相似地，Android系统是一个Activity 的 onCreate() 方法调用开始启动程序。一个回调方法 - 启动一个活动，以及其它回调方法，如销毁一个活动，活动的生命周期如下图所示序列：



Activity类定义了以下的回调方法，即事件。并不需要实现所有的回调方法。然而重要的是了解每一个变化以及实现，以确保应用程序如用户所期望的行为或方式。

回调方法	描述
onCreate()	这是第一次回调，活动在第一次创建时调用。
onStart()	这个回调被称为活动时变成对用户可见。
onResume()	这就是所谓的启动，当用户与应用程序交互。
onPause()	暂停活动不接收用户输入并不执行任何代码并调用时，当前的活动被暂停，恢复以前的活动。
onStop()	这个回调被称为活动时不再可见
onDestroy()	活动前由系统被销毁，调用此回调
onRestart()	活动重新启动时，停止后调用此回调

例子

这个例子通过简单的步骤，显示Android应用程序活动的生命周期。按照下面的步骤来修改Android应用程序，在创建的 Hello World 范例章节：

步骤	描述
1	我们将使用Eclipse IDE中创建一个Android应用程序，并将其命名在HelloWorld 的包下，如：com.example.helloworld的Hello World范例章节解释。
2	修改Main活动文件MainActivity.java,其余文件保持不变。
3	运行该应用程序启动Android模拟器并验证应用程序中所做的更改结果。

以下是修改主要活动文件src/com.example.helloworld/MainActivity.java后的内容，该文件包含了每个基本生命周期方法。 Log.d() 方法是用来生成日志消息：

```
package com.example.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.util.Log;

public class MainActivity extends Activity {
   String msg = "Android : ";

   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Log.d(msg, "The onCreate() event");
   }

   /** Called when the activity is about to become visible. */
   @Override
   protected void onStart() {
      super.onStart();
      Log.d(msg, "The onStart() event");
   }

   /** Called when the activity has become visible. */
   @Override
   protected void onResume() {
      super.onResume();
      Log.d(msg, "The onResume() event");
   }

   /** Called when another activity is taking focus. */
   @Override
   protected void onPause() {
      super.onPause();
```

```
        Log.d(msg, "The onPause() event");
    }

    /** Called when the activity is no longer visible. */
    @Override
    protected void onStop() {
        super.onStop();
        Log.d(msg, "The onStop() event");
    }

    /** Called just before the activity is destroyed. */
    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.d(msg, "The onDestroy() event");
    }
}
```

活动类加载UI组件，使用在项目的 res/layout 文件夹的XML文件。下面的语句从 res/layout/activity_main.xml 文件加载UI组件：

```
setContentView(R.layout.activity_main);
```

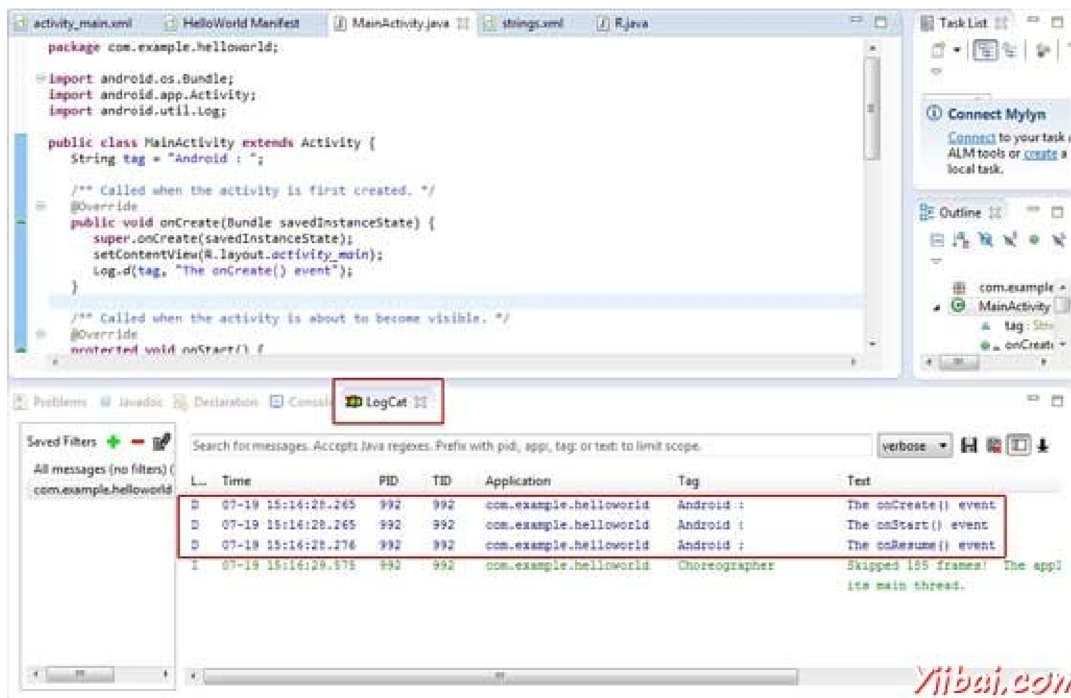
应用程序可以有一个或多个活动受任何限制。每个活动定义为应用程序必须在AndroidManifest.xml文件中声明，必须声明应用程序的主要活动列表<intent-filter>，包括如下**MAIN** 操作和 **LAUNCHER** 类：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/androi
    package="com.example.helloworld"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" /
                <category android:name="android.intent.category.LAU
            </intent-filter>
        </activity>
    </application>
</manifest>
```

如果 MAIN 动作或LAUNCHER类不声明一个活动，那么应用程序图标将不会出现在应用程序列表的主屏幕上。

现在尝试运行修改后的 Hello World！应用程序。假设已经创建了AVD，同时做环境设置。从Eclipse运行应用程序，打开一个项目的活动文件，并单击“**Run**”图标。Eclipse 在 AVD上安装应用程序，并启动它，如果设置和应用都没有问题，它会显示仿真器窗口中，也应该看到以下日志消息在Eclipse IDE LogCat 窗口：

```
07-19 15:00:43.405: D/Android :(866): The onCreate() event
07-19 15:00:43.405: D/Android :(866): The onStart() event
07-19 15:00:43.415: D/Android :(866): The onResume() event
```



点击红色按钮 - 在Android模拟器上的红色按钮，它会在Eclipse IDE窗口LogCat中生成以下事件消息：

```
07-19 15:01:10.995: D/Android :(866): The onPause() event
07-19 15:01:12.705: D/Android :(866): The onStop() event
```

让我们再次尝试单击菜单按钮 - Android菜单按钮在Android模拟器中，它会在Eclipse IDE 的 LogCat窗口中生成以下事件消息：

```
07-19 15:01:13.995: D/Android :(866): The onStart() event
07-19 15:01:14.705: D/Android :(866): The onResume() event
```

接下来，就让我们再次尝试点击后退按钮 - Android后退按钮在Android模拟器，它会生成以下事件消息在Eclipse IDE LogCat窗口中，以上完成Android应用程序Acitivity 的生命周期。

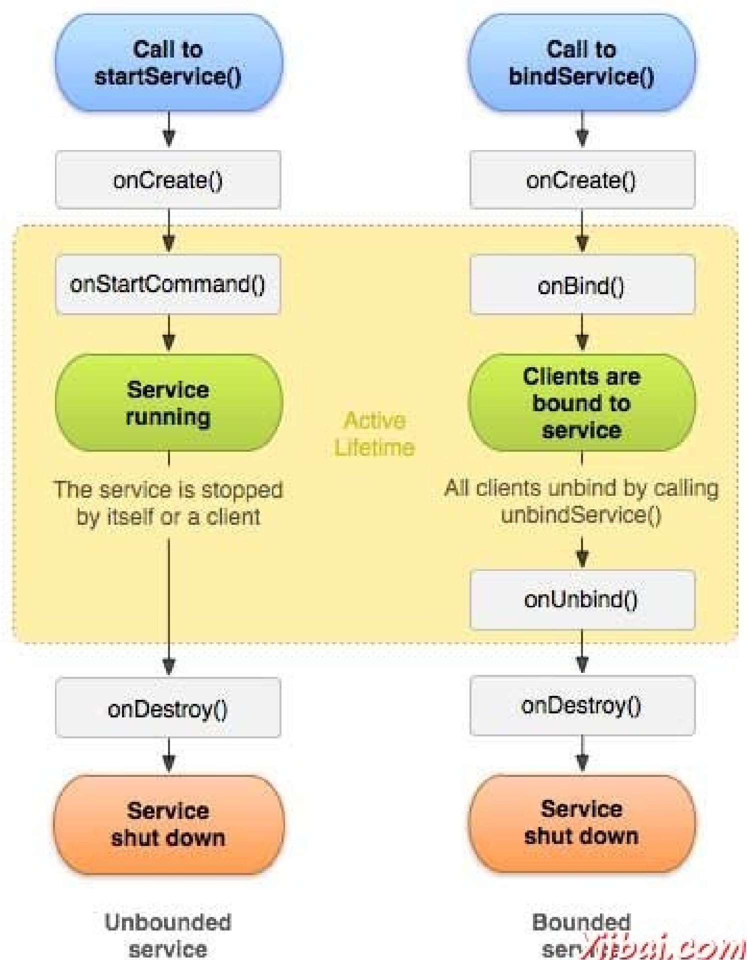
```
07-19 15:33:15.687: D/Android :(992): The onPause() event
07-19 15:33:15.525: D/Android :(992): The onStop() event
07-19 15:33:15.525: D/Android :(992): The onDestroy() event
```

Android Service - Android开发教程

Service(服务)是一种在后台运行，执行长时间运行的操作，无需与用户交互的组件。例如，一个服务可以在后台播放音乐，用户在不同的应用程序或者可能通过网络获取数据，而不阻塞用户交互活动。本质上，一个服务可以采取两种状态：

状态	描述
Started	当一个应用程序组件，如活动，开始通过调用StartService()启动一个服务。开始以后服务可以无限期地在后台运行，即使启动它的组件被破坏。
Bound	当一个应用程序组件绑定调用bindService()方法绑定服务。绑定服务提供客户端 - 服务器的接口，允许组件进行交互的服务，发送请求，得到结果，这样可以跨进程进程间通信（IPC）。

每个服务都具有生命周期回调方法，可以实现监视服务的状态变化，并在适当的阶段执行工作。下图左侧显示的整个生命周期由StartService()创建提供服务，右边的图显示bindService()创建的整个生命周期提供服务：



要创建一个服务，需要创建一个Java类，扩展Service基类或者它的子类。Service基类定义各种回调方法，如下面表格给出。但是也并不需要实现所有的回调方法。重要的是要了解每一个变化以及实现，以确保应用程序能如用户所期望的行为方式运行。

回调	描述
onStartCommand()	系统调用此方法当另一组件，如一个活动，通过调用startService()要求该服务启动。如果要实现方法，它工作完成后停止服务，通过调用stopSelf()或stopService()方法。
onBind()	该系统调用这个方法当其他组件要通过调用bindService()绑定服务。如果实现此方法，必须提供客户端与服务进行通信，通过返回一个IBinder对象的接口。必须实现此方法，但如果不希望被绑定，那么应该返回null。
onUnbind()	系统调用此方法，当所有客户都从服务发布的特定接口断开。
onRebind()	该系统调用这个方法时，新的客户端已连接到服务，它事先未通知，所有已经上解除绑定后（意向）断开它。
onCreate()	该系统调用时，使用onStartCommand()或onBind()首先创建的服务这个方法。此调用需要执行一次性安装。
onDestroy()	系统调用这个方法当服务不再使用（被销毁）。服务应该实现这个用于清理，如线程，注册的侦听器，接收器等任何资源

下面的主服务演示每一个方法生命周期：

```
package com.yiibai;

import android.app.Service;
import android.os.IBinder;
import android.content.Intent;
import android.os.Bundle;

public class HelloService extends Service {

   /** indicates how to behave if the service is killed */
   int mStartMode;
   /** interface for clients that bind */
   IBinder mBinder;
   /** indicates whether onRebind should be used */
   boolean mAllowRebind;

   /** Called when the service is being created. */
   @Override
```

```
    public void onCreate() {

    }

    /** The service is starting, due to a call to startService() */
    @Override
    public int onStartCommand(Intent intent, int flags, int startId)
       return mStartMode;
    }

    /** A client is binding to the service with bindService() */
    @Override
    public IBinder onBind(Intent intent) {
       return mBinder;
    }

    /** Called when all clients have unbound with unbindService() */
    @Override
    public boolean onUnbind(Intent intent) {
       return mAllowRebind;
    }

    /** Called when a client is binding to the service with bindServ
    @Override
    public void onRebind(Intent intent) {

    }

    /** Called when The service is no longer used and is being destr
    @Override
    public void onDestroy() {

    }
 }
```

示例

这个例子将通过简单的步骤显示了如何创建Android服务。按照下面的步骤来修改前面章节创建的Android应用程序 - Hello World示例 ：

步骤	描述
1	使用Eclipse IDE创建Android应用程序，并将其命名为HelloWorld在包com.example.helloworld下， 类似Hello World示例章节中一样。
2	修改主要活动文件MainActivity.java添加startService()和stopService()方法。
3	在包com.example.helloworld下创建一个新的Java文件MyService.java。该文件将有实现Android服务相关的方法。
4	使用 <service.../>标签定义AndroidManifest.xml文件服务。一个应用可以有一个或多个服务，没有任何限制。
5	修改res/layout/activity_main.xml文件的默认内容包括线性布局中的两个按钮。
6	定义两个常量start_service和stop_service在 res/values/strings.xml 文件中
7	运行该应用程序启动Android模拟器并验证应用程序所做的修改结果。

以下是改性主要活动文件 **src/com.example.helloworld/MainActivity.java** 的内容。这个文件包括每个基本的生命周期方法。添加 **StartService()** 和 **stopService()** 方法来启动和停止服务。

```
package com.example.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.content.Intent;
import android.view.View;

public class MainActivity extends Activity {

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.activity_main, menu);
      return true;
   }

   // Method to start the service
   public void startService(View view) {
      startService(new Intent(getBaseContext(), MyService.class));
   }

   // Method to stop the service
   public void stopService(View view) {
      stopService(new Intent(getBaseContext(), MyService.class));
   }
}
```

以下是**src/com.example.helloworld/MyService.java** 的内容。这个文件可以有一个或多个方法来使用服务。现在要实现只有两个方法 onStartCommand() 和 onDestroy() ：

```
package com.example.helloworld;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.widget.Toast;

public class MyService extends Service {
   @Override
   public IBinder onBind(Intent arg0) {
      return null;
   }

   @Override
   public int onStartCommand(Intent intent, int flags, int startId
      // Let it continue running until it is stopped.
      Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).sh
      return START_STICKY;
   }
   @Override
   public void onDestroy() {
      super.onDestroy();
      Toast.makeText(this, "Service Destroyed", Toast.LENGTH_LONG)
   }
}
```

下面将 **AndroidManifest.xml** 文件的内容修改。在这里添加 **<service.../>** 标签，包括服务：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/androic
   package="com.example.helloworld"
   android:versionCode="1"
   android:versionName="1.0" >
   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="15" />
   <application
       android:icon="@drawable/ic_launcher"
       android:label="@string/app_name"
       android:theme="@style/AppTheme" >
       <activity
           android:name=".MainActivity"
           android:label="@string/title_activity_main" >
           <intent-filter>
               <action android:name="android.intent.action.MAIN" />
               <category android:name="android.intent.category.LAUN
           </intent-filter>
       </activity>
       <service android:name=".MyService" />
   </application>
</manifest>
```

将以下是 **res/layout/activity_main.xml** 文件的内容，包括两个按钮：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <Button android:id="@+id/btnStartService"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/start_service"
   android:onClick="startService"/>

   <Button android:id="@+id/btnStopService"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/stop_service"
   android:onClick="stopService" />

</LinearLayout>
```

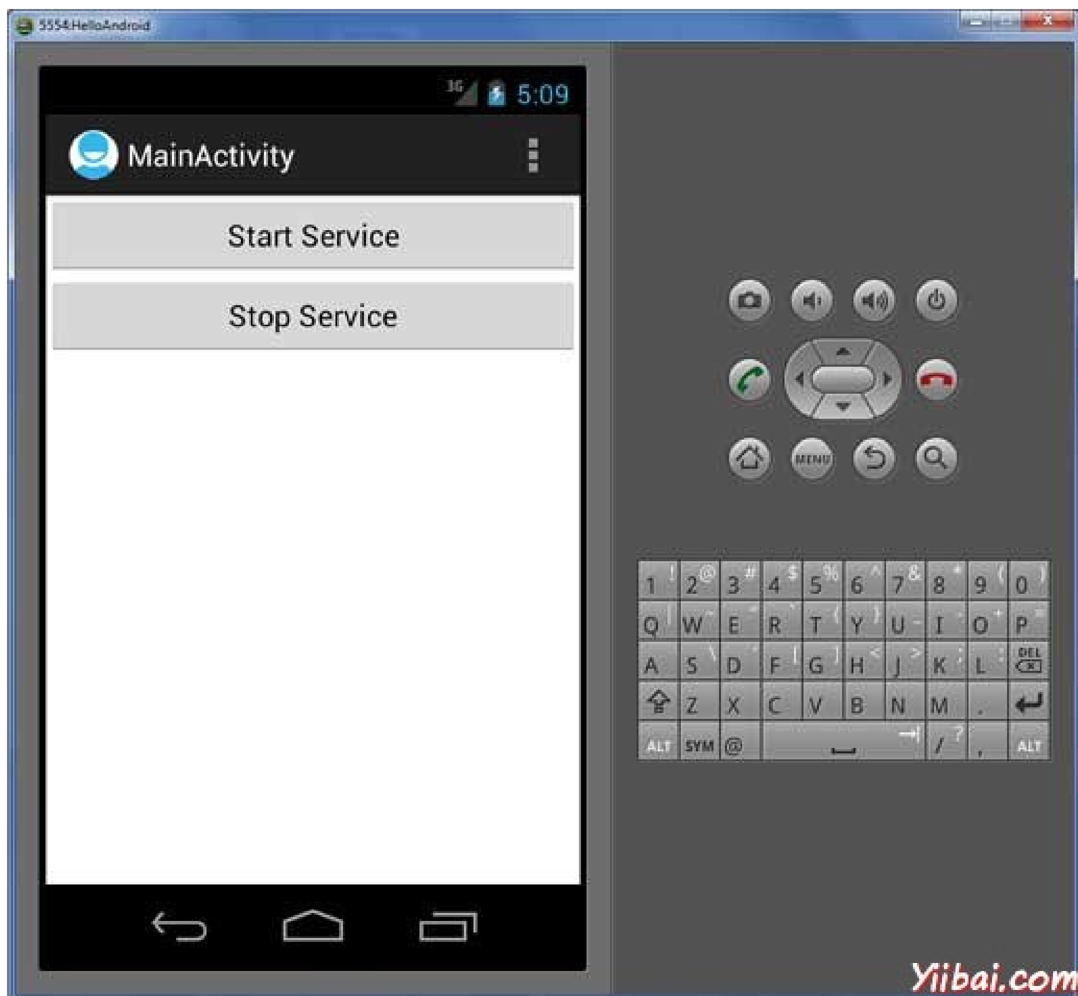
下面将在 **res/values/strings.xml** 中定义两个新的常量：

```
<resources>

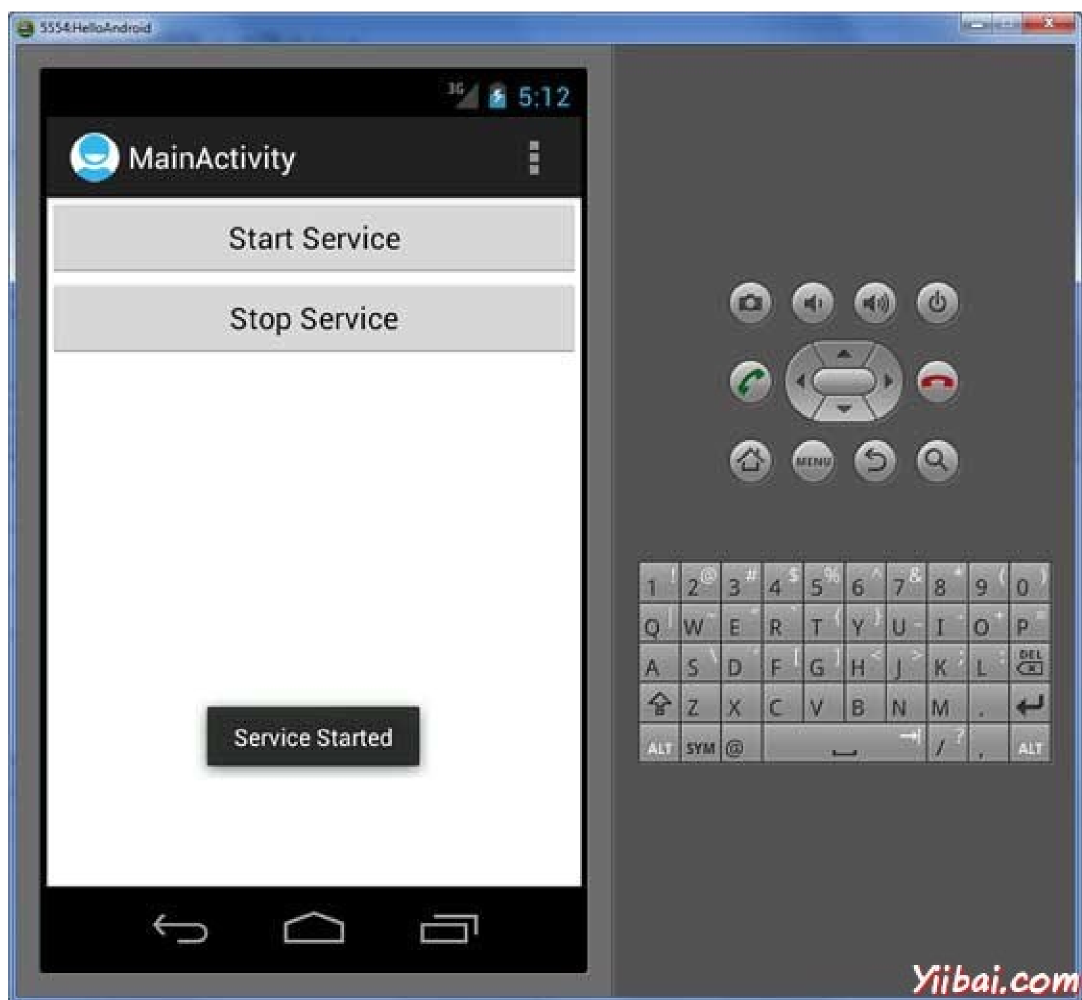
    <string name="app_name">HelloWorld</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>
    <string name="start_service">Start Service</string>
    <string name="stop_service">Stop Service</string>

</resources>
```

现在运行修改后的 Hello World！应用程序。假设创建了AVD 并同时做了环境设置。要在Eclipse运行的应用程序，打开一个项目的活动文件，从工具栏上找到并单击 “**run**” 图标。 Eclipse AVD上安装的应用程序，并启动它，如果一切设置以及应用都没有问题，那么将会显示以下模拟器窗口：



要开始服务，现在就点击启动服务按钮，onStartCommand() 方法在程序中，每一个服务开始后将出现消息在模拟器底部，如下：



要停止该服务，可以点击停止服务（Stop Service）按钮。

Android广播接收器 - Android开发教程

广播接收器(Broadcast)简单地从其他应用程序或系统响应广播消息。这些消息有时称为事件或意图。例如，应用程序也可以发起广播，以让其他应用程序知道某些数据已经被下载到设备上，可供它们使用。广播接收器会拦截此通信，并会采取适当操作（动作）。

以下两个重要的步骤，在使用广播接收器工作系统及广播意图：

- 创建广播接收器
- 注册广播接收器

还有一个附加的步骤，要实现自定义的意图，那么将必须创建并广播意图。

创建广播接收器

实现广播接收机BroadcastReceiver类的一个子类并重写 onReceive()方法，其中每个收到消息作为一个 Intent 对象参数。

```
public class MyReceiver extends BroadcastReceiver {

   @Override
   public void onReceive(Context context, Intent intent) {
      Toast.makeText(context, "Intent Detected.", Toast.LENGTH_LONG
   }

}
```

注册广播接收器

应用程序侦听特定的广播意图是通过在 AndroidManifest.xml 文件中注册一个广播接收器。寄存器 MyReceiver 系统生成事件 ACTION_BOOT_COMPLETED，在Android系统完成了启动过程后，这是由系统启动执行的。

```
<application
   android:icon="@drawable/ic_launcher"
   android:label="@string/app_name"
   android:theme="@style/AppTheme" >

   <receiver android:name="MyReceiver">
      <intent-filter>
         <action android:name="android.intent.action.BOOT_COMPLETED
      </action>
      </intent-filter>
   </receiver>

</application>
```

当 Android 设备启动，它会被截获 **BroadcastReceiver** 的 **MyReceiverand** 内实现逻辑，首先 **onReceive**() 将被执行。

有几个系统产生的事件定义在最后意图类的静态字段。下表列出了一些重要的系统事件。

事件常量	描述
android.intent.action.BATTERY_CHANGED	持久广播含充电状态，级别，以及其他相关的电池信息。
android.intent.action.BATTERY_LOW	显示设备的电池电量低。
android.intent.action.BATTERY_OKAY	指示电池正在低点后但没有问题。
android.intent.action.BOOT_COMPLETED	一次播出后，系统已完成启动。
android.intent.action.BUG_REPORT	显示活动报告的错误。
android.intent.action.CALL	执行呼叫由数据指定某人。
android.intent.action.CALL_BUTTON	用户按下“呼叫”按钮进入拨号器或其他适当的用户界面发出呼叫。
android.intent.action.DATE_CHANGED	日期改变。
android.intent.action.REBOOT	有设备重启。

广播定制意图

如果希望应用程序本身生成并发送自定义意图，那么必须使用sendBroadcast()方法里面活动类来创建和发送这些的意图。使用（意向）sendStickyBroadcast() 方法意图是粘粘的，这意味着所发送的意图保持周广围播出后完成。

```
public void broadcastIntent(View view)
{
   Intent intent = new Intent();
   intent.setAction("com.yiibai.CUSTOM_INTENT");
   sendBroadcast(intent);
}
```

意图 com.yiibai.CUSTOM_INTENT也可以以注册类似的方式，因为我们产生注册系统的意图。

```
<application
   android:icon="@drawable/ic_launcher"
   android:label="@string/app_name"
   android:theme="@style/AppTheme" >

   <receiver android:name="MyReceiver">
      <intent-filter>
         <action android:name="com.yiibai.CUSTOM_INTENT">
      </action>
      </intent-filter>
   </receiver>

</application>
```

示例

这个例子将解释如何创建BroadcastReceiver 拦截自定义意图。熟悉自定义意图后，就可以编写应用程序来拦截系统生成的意图。现在按照下面的步骤来修改前面创建的Hello World范例中 Android 应用程序：

步骤	描述
1	使用Eclipse IDE创建Android应用程序，并将其命名为HelloWorld在包com.example.helloworld下，类似Hello World示例章节中一样。
2	修改主要活动文件MainActivity.java添加broadcastIntent()方法。
3	在包com.example.helloworld下创建一个新的Java文件MyReceiver.java，并定义一个BroadcastReceiver。
4	应用程序可以处理一个或多个自定义和系统的意图不受任何限制。要拦截每一个意图，必须使用 <receiver.../>标签并注册在AndroidManifest.xml文件中。
5	修改 res/layout/activity_main.xml 文件的默认内容包括：一个按钮广播意图。
6	定义常量 broadcast_inte 在 ntres/values/strings.xml文件中
7	运行该应用程序启动Android模拟器并验证应用程序所做的修改结果。

以下是修改主要活动文件 **src/com.example.helloworld/MainActivity.java** 后的内容。这个文件包括每个生命周期方法。这里添加了 **broadcastIntent()** 方法来广播自定义的意图。

```
package com.example.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.content.Intent;
import android.view.View;

public class MainActivity extends Activity {

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.activity_main, menu);
      return true;
   }
   // broadcast a custom intent.
   public void broadcastIntent(View view)
   {
      Intent intent = new Intent();
      intent.setAction("com.yiibai.CUSTOM_INTENT");
      sendBroadcast(intent);
   }
}
```

下面是 **src/com.example.helloworld/MyReceiver.java** 的内容:

```
package com.example.helloworld;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class MyReceiver extends BroadcastReceiver {

   @Override
   public void onReceive(Context context, Intent intent) {
      Toast.makeText(context, "Intent Detected.", Toast.LENGTH_LONG
   }

}
```

下面将 AndroidManifest.xml 文件的内容修改。在这里添加 <service.../>标签，包括服务：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/androic
   package="com.example.helloworld"
   android:versionCode="1"
   android:versionName="1.0" >
   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="15" />
   <application
       android:icon="@drawable/ic_launcher"
       android:label="@string/app_name"
       android:theme="@style/AppTheme" >
       <activity
           android:name=".MainActivity"
           android:label="@string/title_activity_main" >
           <intent-filter>
               <action android:name="android.intent.action.MAIN" /:
               <category android:name="android.intent.category.LAUN
           </intent-filter>
       </activity>
       <receiver android:name="MyReceiver">
          <intent-filter>
          <action android:name="com.yiibai.CUSTOM_INTENT">
          </action>
          </intent-filter>
      </receiver>
   </application>
</manifest>
```

以下将 **res/layout/activity_main.xml** 文件的内容包括一个按钮来广播自定义意图：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <Button android:id="@+id/btnStartService"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/broadcast_intent"
   android:onClick="broadcastIntent"/>

</LinearLayout>
```

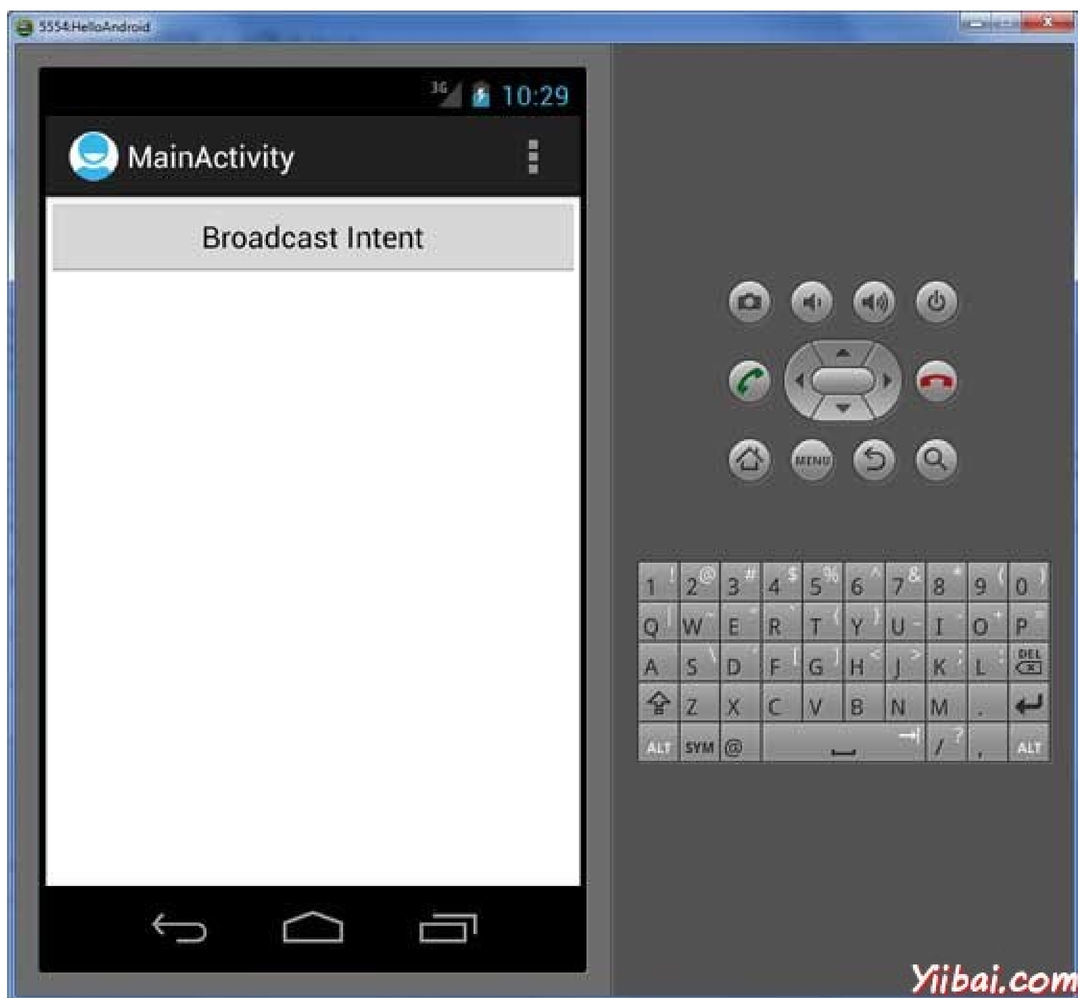
下面将在 res/values/strings.xml 中定义两个新的常量的内容：

```
<resources>

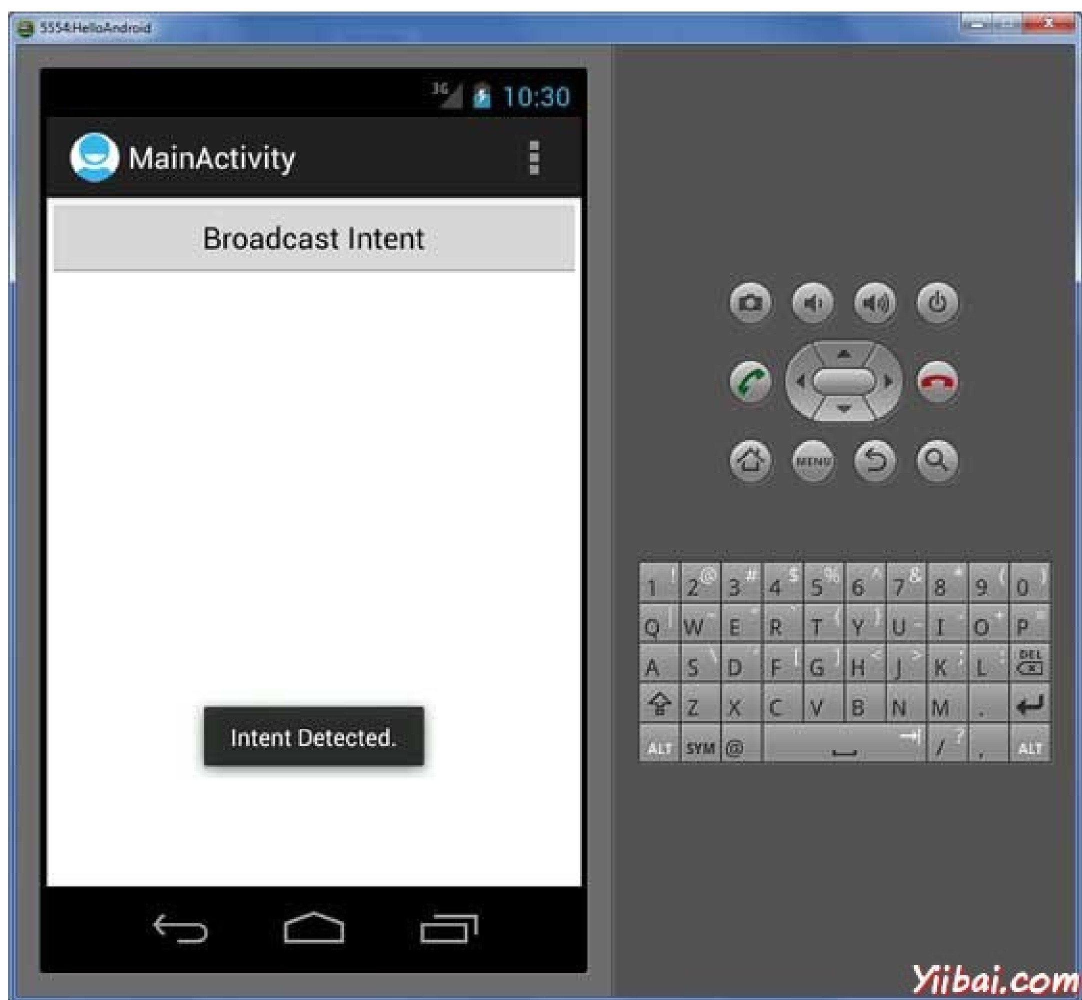
    <string name="app_name">HelloWorld</string>
    <string name="hello_world">Hello world!</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>
    <string name="broadcast_intent">Broadcast Intent</string>

</resources>
```

现在运行修改后的 Hello World！应用程序。假设创建了AVD并设置了环境。要从Eclipse运行的应用程序，首先打开一个项目的活动文件，从工具栏单击“run” 图标。 Eclipse AVD安装的应用程序，并启动它，如果设置和应用都没有问题，将会显示以下模拟器窗口：



现在广播自定义的意图，点击上广播意图按钮，这将广播自定义在 “com.yiibai.CUSTOM_INTENT” 注册BroadcastReceiver 的意图将被 MyReceiver 拦截。 实现的逻辑如下出现底部的模拟器：



可以尝试执行其他 BroadcastReceiver 拦截系统的意图，如系统启动，更改日期，电池电量不足等。

Android内容提供者 - Android开发教程

内容提供程序（Provider）组件从一个应用到其他请求提供数据。通过 ContentResolver 类的方法这样的请求处理。内容提供程序使用不同的方式来存储数据，并且可以将数据存储在数据库中，文件中，甚至在网络上。

每一个 Android 应用程序运行在自己的进程保持一个应用程序数据，在另外一个应用程序中隐藏自己的权限。但有时需要在应用程序之间共享数据。这时内容提供程序是非常有用。

内容提供程序将内容集中在一个地方，让许多不同的应用访问。内容提供程序的行非常像数据库，可以对它进行查询，编辑等操作，添加或删除可使用 insert(), update(), delete(), query() 方法。在大多数情况下，这些数据都存储在SQlite数据库。

内容提供程序实施 ContentProvider 类的子类，必须实现了一套标准的 API，使其他应用程序来执行事务。

```
public class MyContentProvider extends  ContentProvider {

}
```

内容的URI

要查询内容提供程序，可以指定 URI 形式如以下格式的查询字符串：

```
<prefix>://<authority>/<data_type>/<id>
```

这里是URI的各个部分的细节

部分	描述
prefix	始终设置内容为 ://
authority	规定内容提供商的名称，例如联系人，浏览器等。对于第三方内容提供商，这可能是完全合格的名称，如 *com.yiibai.statusprovider*
data_type	表示数据，特定提供程序提供的类型。例如，如果得到所有的联系人的通讯录内容提供程序，那么数据路径URI是这样的 *content://contacts/people*
id	规定要求的特定记录。例如，如果正在寻找联系人编号为5，在联系人内容提供者中，则URI是这样的 *content://contacts/people/5.*

创建内容提供者

以下是简单的步骤用来创建自己的内容提供者的数量。

- 首先，需要创建一个内容提供者扩展 ContentProvider 基类。
- 其次，需要定义内容提供者用于访问内容的 URI 地址。
- 接下来，需要创建自己的数据库用于保存内容。通常情况下，Android使用SQLite数据库，并且框架需要重写 onCreate() 方法会使用 SQLite开放的Helper方法来创建或打开提供者数据库。当启动应用程序时，每个内容提供者的onCreate()方法调用处理程序在主应用程序。
- 接下来，必须实现内容提供者查询来执行不同的数据库的具体操作。
- 最后，在activity文件使用<provider>标签注册内容提供者。

下面是需要覆盖内容提供程序类的方法的列表：

- onCreate() 方法被称为提供者开始。
- query() 方法接收来自客户端的请求。返回的结果作为一个Cursor对象。
- insert() 方法插入一条新记录到内容提供者。
- delete() 方法从内容提供者删除记录。
- update() 方法从内容提供者更新现有记录。
- getType() 此方法在给定的URI返回 MIME 类型的数据。

示例

这个例子将解释如何创建自己的 ContentProvider。因此按照下面的步骤类似于我们之前创建Hello World范例：

Step	描述
1	使用Eclipse IDE创建Android应用程序，并将它命名为MyContentProviderunder在包com.example.mycontentprovider下并使用空的Activity。
2	修改主要活动文件MainActivity.java增加两个新的方法onClickAddName() 和 onClickRetrieveStudents()。
3	创建一个新的名为StudentsProvider.java的java文件在packagecom.example.mycontentprovider包下，并定义实际提供者和相关方法。
4	使用注册内容提供者在AndroidManifest.xml文件中的<provider.../>标签
5	修改res/layout/activity_main.xml文件的默认内容包括一个小的GUI添加学生记录。
6	在res/values/strings.xml文件中定义所需的常量
7	运行该应用程序启动Android模拟器和验证应用程序所做的修改结果。

以下是主活动文件 **src/com.example.mycontentprovider/MainActivity.java** 修改后的内容。这个文件可以包括每个生命周期方法。我们已经增加了两个新方法 **onClickAddName()** 和 **onClickRetrieveStudents()** 来处理用户与应用程序交互。

```
package com.example.mycontentprovider;

import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.content.ContentValues;
import android.content.CursorLoader;
import android.database.Cursor;
import android.view.Menu;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
   }
```

```
   public void onClickAddName(View view) {
      // Add a new student record
      ContentValues values = new ContentValues();

      values.put(StudentsProvider.NAME,
      ((EditText)findViewById(R.id.txtName)).getText().toString());

      values.put(StudentsProvider.GRADE,
      ((EditText)findViewById(R.id.txtGrade)).getText().toString()

      Uri uri = getContentResolver().insert(
      StudentsProvider.CONTENT_URI, values);

      Toast.makeText(getBaseContext(),
      uri.toString(), Toast.LENGTH_LONG).show();
   }

   public void onClickRetrieveStudents(View view) {
      // Retrieve student records
      String URL = "content://com.example.provider.College/students
      Uri students = Uri.parse(URL);
      Cursor c = managedQuery(students, null, null, null, "name");
      if (c.moveToFirst()) {
         do{
            Toast.makeText(this,
            c.getString(c.getColumnIndex(StudentsProvider._ID)) +
            ", " +  c.getString(c.getColumnIndex( StudentsProvider.
            ", " + c.getString(c.getColumnIndex( StudentsProvider.(
            Toast.LENGTH_SHORT).show();
         } while (c.moveToNext());
      }
   }
}
```

在 com.example.mycontentprovider 包下创建新的 StudentsProvider.java 文件，以下是内容：

```
package com.example.mycontentprovider;

import java.util.HashMap;

import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
```

```
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteQueryBuilder;
import android.net.Uri;
import android.text.TextUtils;

public class StudentsProvider extends ContentProvider {

   static final String PROVIDER_NAME = "com.example.provider.Colleg
   static final String URL = "content://" + PROVIDER_NAME + "/stude
   static final Uri CONTENT_URI = Uri.parse(URL);

   static final String _ID = "_id";
   static final String NAME = "name";
   static final String GRADE = "grade";

   private static HashMap<String, String> STUDENTS_PROJECTION_MAP;

   static final int STUDENTS = 1;
   static final int STUDENT_ID = 2;

   static final UriMatcher uriMatcher;
   static{
      uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
      uriMatcher.addURI(PROVIDER_NAME, "students", STUDENTS);
      uriMatcher.addURI(PROVIDER_NAME, "students/#", STUDENT_ID);
   }

   /**
    * Database specific constant declarations
    */
   private SQLiteDatabase db;
   static final String DATABASE_NAME = "College";
   static final String STUDENTS_TABLE_NAME = "students";
   static final int DATABASE_VERSION = 1;
   static final String CREATE_DB_TABLE =
      " CREATE TABLE " + STUDENTS_TABLE_NAME +
      " (_id INTEGER PRIMARY KEY AUTOINCREMENT, " +
      " name TEXT NOT NULL, " +
      " grade TEXT NOT NULL);";

   /**
    * Helper class that actually creates and manages
    * the provider's underlying data repository.
    */
   private static class DatabaseHelper extends SQLiteOpenHelper {
       DatabaseHelper(Context context){
          super(context, DATABASE_NAME, null, DATABASE_VERSION);
       }

       @Override
       public void onCreate(SQLiteDatabase db)
       {
          db.execSQL(CREATE_DB_TABLE);
```

```
      }

      @Override
      public void onUpgrade(SQLiteDatabase db, int oldVersion,
                           int newVersion) {
         db.execSQL("DROP TABLE IF EXISTS " +  STUDENTS_TABLE_NAME
         onCreate(db);
      }
   }

   @Override
   public boolean onCreate() {
      Context context = getContext();
      DatabaseHelper dbHelper = new DatabaseHelper(context);
      /**
       * Create a write able database which will trigger its
       * creation if it doesn't already exist.
       */
      db = dbHelper.getWritableDatabase();
      return (db == null)? false:true;
   }

   @Override
   public Uri insert(Uri uri, ContentValues values) {
      /**
       * Add a new student record
       */
      long rowID = db.insert(    STUDENTS_TABLE_NAME, "", values);
      /**
       * If record is added successfully
       */
      if (rowID > 0)
      {
         Uri _uri = ContentUris.withAppendedId(CONTENT_URI, rowID);
         getContext().getContentResolver().notifyChange(_uri, null)
         return _uri;
      }
      throw new SQLException("Failed to add a record into " + uri);
   }

   @Override
   public Cursor query(Uri uri, String[] projection, String selecti
                       String[] selectionArgs, String sortOrder) {

      SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
      qb.setTables(STUDENTS_TABLE_NAME);

      switch (uriMatcher.match(uri)) {
      case STUDENTS:
         qb.setProjectionMap(STUDENTS_PROJECTION_MAP);
         break;
      case STUDENT_ID:
         qb.appendWhere( _ID + "=" + uri.getPathSegments().get(1));
```

```
         break;
      default:
         throw new IllegalArgumentException("Unknown URI " + uri);
      }
      if (sortOrder == null || sortOrder == ""){
         /**
          * By default sort on student names
          */
         sortOrder = NAME;
      }
      Cursor c = qb.query(db,    projection,    selection, selectio
                   null, null, sortOrder);
      /**
       * register to watch a content URI for changes
       */
      c.setNotificationUri(getContext().getContentResolver(), uri);

      return c;
   }

   @Override
   public int delete(Uri uri, String selection, String[] selection
      int count = 0;

      switch (uriMatcher.match(uri)){
      case STUDENTS:
         count = db.delete(STUDENTS_TABLE_NAME, selection, selectio
         break;
      case STUDENT_ID:
         String id = uri.getPathSegments().get(1);
         count = db.delete( STUDENTS_TABLE_NAME, _ID +  " = " + id
               (!TextUtils.isEmpty(selection) ? " AND (" +
               selection + ')' : ""), selectionArgs);
         break;
      default:
         throw new IllegalArgumentException("Unknown URI " + uri);
      }

      getContext().getContentResolver().notifyChange(uri, null);
      return count;
   }

   @Override
   public int update(Uri uri, ContentValues values, String selectio
                 String[] selectionArgs) {
      int count = 0;

      switch (uriMatcher.match(uri)){
      case STUDENTS:
         count = db.update(STUDENTS_TABLE_NAME, values,
                selection, selectionArgs);
         break;
      case STUDENT_ID:
```

```
         count = db.update(STUDENTS_TABLE_NAME, values, _ID +
                 " = " + uri.getPathSegments().get(1) +
                 (!TextUtils.isEmpty(selection) ? " AND (" +
                 selection + ')' : ""), selectionArgs);
         break;
      default:
         throw new IllegalArgumentException("Unknown URI " + uri );
      }
      getContext().getContentResolver().notifyChange(uri, null);
      return count;
   }

   @Override
   public String getType(Uri uri) {
      switch (uriMatcher.match(uri)){
      /**
       * Get all student records
       */
      case STUDENTS:
         return "vnd.android.cursor.dir/vnd.example.students";
      /**
       * Get a particular student
       */
      case STUDENT_ID:
         return "vnd.android.cursor.item/vnd.example.students";
      default:
         throw new IllegalArgumentException("Unsupported URI: " + u
      }
   }
}
```

以下将 **AndroidManifest.xml** 文件的内容修改。在这里添加了<provider.../>标签，包括内容提供者：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/androic
    package="com.example.mycontentprovider"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.mycontentprovider.MainActivit
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" /
                <category android:name="android.intent.category.LAU
            </intent-filter>
        </activity>
        <provider android:name="StudentsProvider"
           android:authorities="com.example.provider.College">
        </provider>
    </application>

</manifest>
```

以下将 **res/layout/activity_main.xml** 文件的内容包括一个按钮来自定义广播意图：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Name" />
    <EditText
    android:id="@+id/txtName"
    android:layout_height="wrap_content"
    android:layout_width="fill_parent" />
    <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Grade" />
    <EditText
    android:id="@+id/txtGrade"
    android:layout_height="wrap_content"
    android:layout_width="fill_parent" />
    <Button
    android:text="Add Name"
    android:id="@+id/btnAdd"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:onClick="onClickAddName" />
    <Button
    android:text="Retrieve Students"
    android:id="@+id/btnRetrieve"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:onClick="onClickRetrieveStudents" />
</LinearLayout>
```

确保 **res/values/strings.xml** 文件有以下内容：

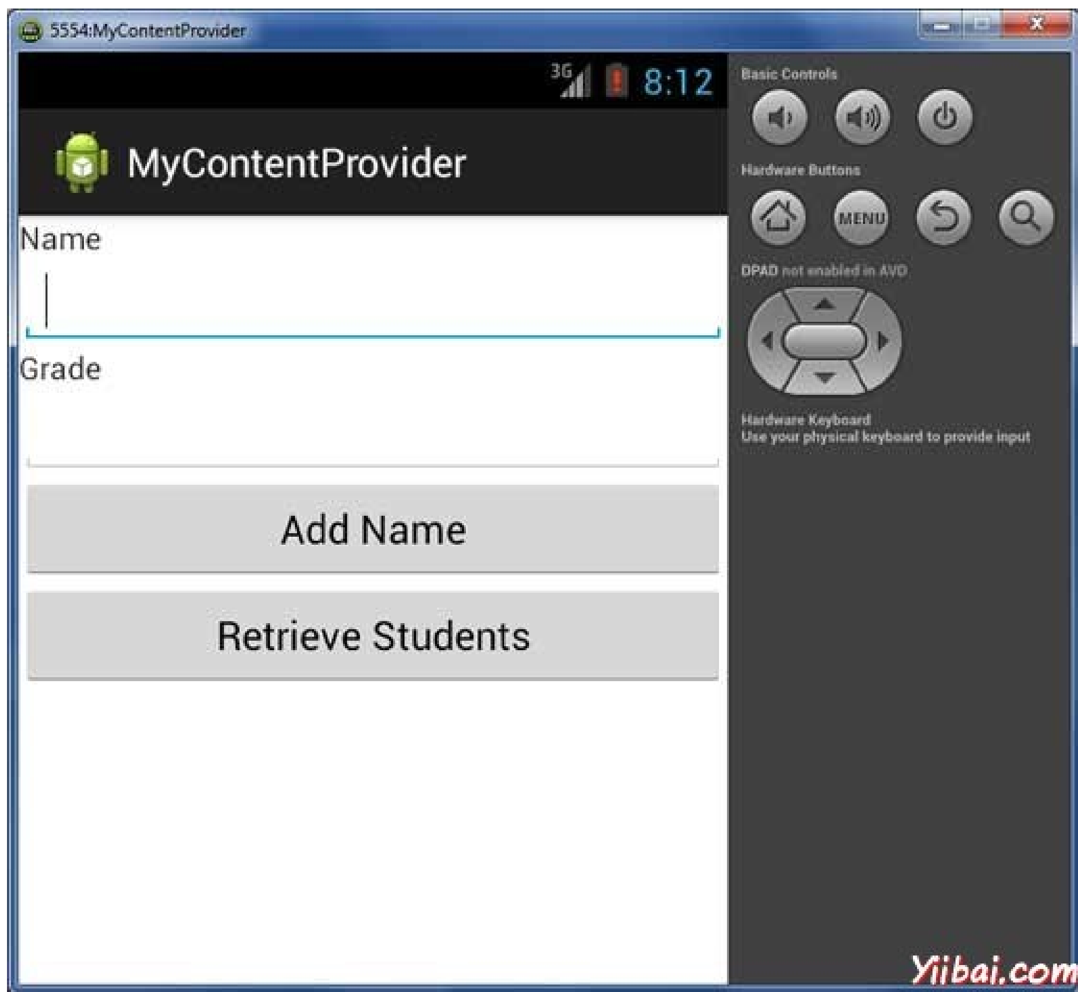
```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">MyContentProvider</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>

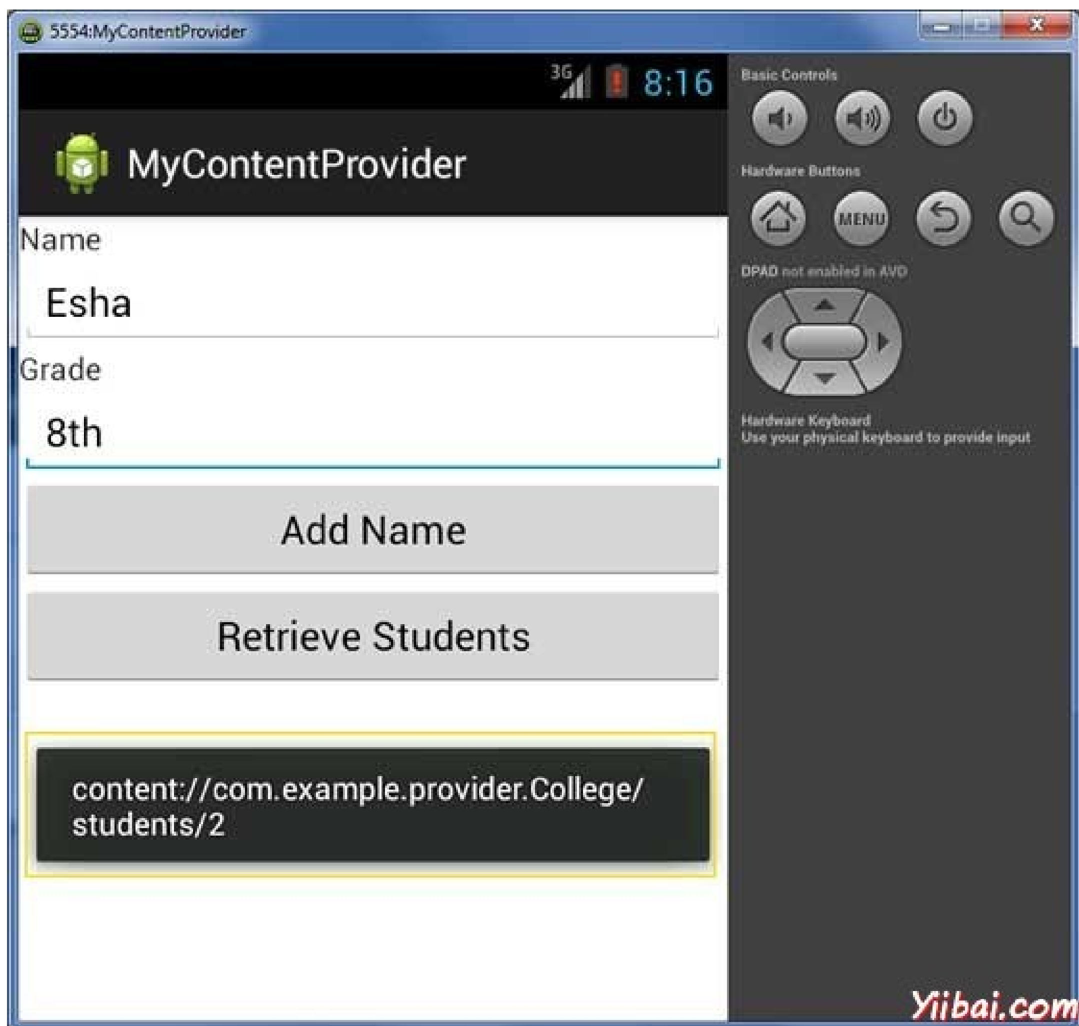
</resources>;
```

现在运行刚刚创建的应用程序 MyContentProvider。假设设置AVD并且已经做好了环境设置。要从Eclipse运行的应用程序，从工具栏打开一个项目的活动文件，并单击“Run” 图标。Eclipse AVD 安装的应用程序后并启动它，如果设置和应用程序

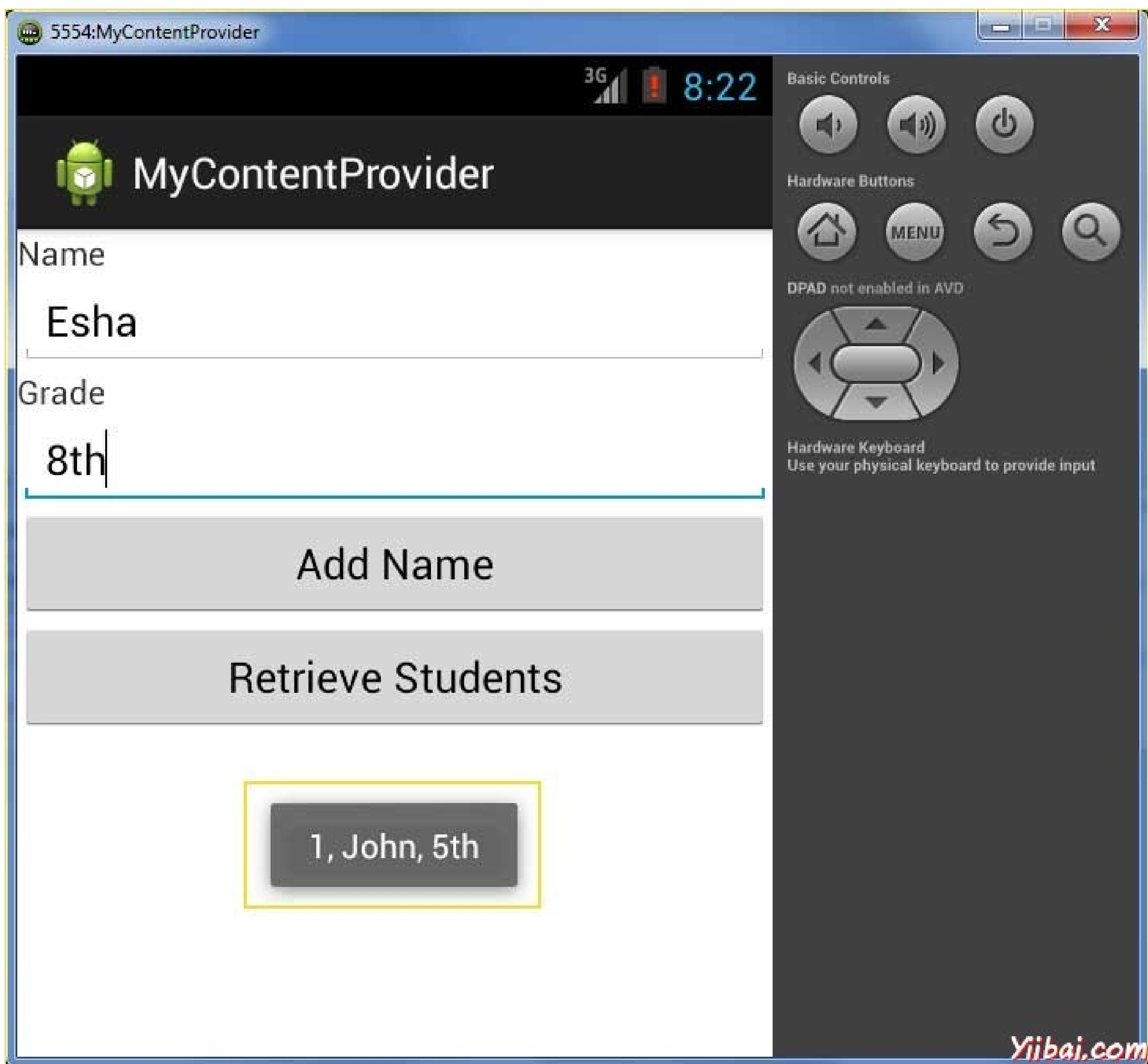
没有问题，它会启动显示仿真器窗口（要耐心，可能有点慢，由你计算机的速度决定）：



现在输入学生姓名和年级，最后单击“Add Name”按钮，将增加学生记录到数据库中，ContentProvider URI也一起添加到数据库中，底部显示的记录数，并会闪烁一条消息。此操作使得我们的insert()方法的使用。让我们重复这个过程，我们的内容提供者在数据库中添加一些更多的学生信息。



正在添加记录在数据库中，现在其时间让ContentProvider来给我们这些记录回，所以让我们的单击检索学生按钮，这将获取并显示所有记录这是按照我们执行query()方法。



可以编写活动 MainActivity.java文件提供的回调函数，然后修改用户界面，更新和删除操作按钮添加和读操作以同样的方式，因为我们已经做了更新和删除操作。

通过这种方式，可以使用现有的内容提供者，如地址簿，或不错的面向数据库应用开发，可以使用内容提供者的概念，可以执行所有的数据库操作，如排序读，写，更新和删除上面的例子中解释。

Android碎片/片段 - Android开发教程

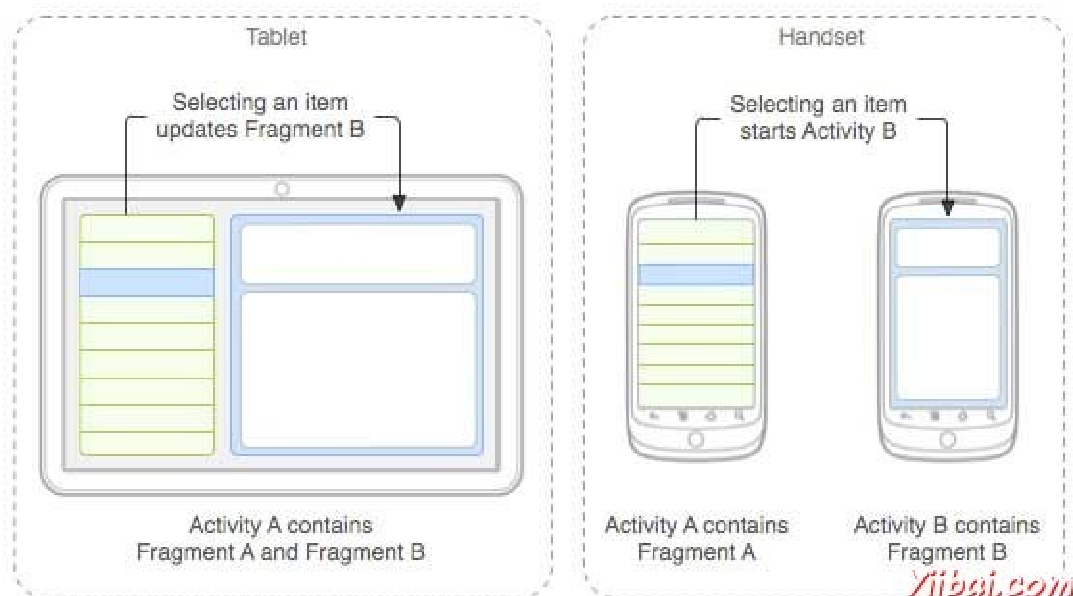
片段(**Fragments**)是一个应用程序的用户界面或行为活动，使活动更加模块化设计，可以放置在一块。一个片段是一种子活动。以下要点有关片段：

- 片段都有自己的布局和规范自己的行为与生命周期回调。
- 可以添加或删除片段在活动而活动运行。
- 可以将多个片段在一个单一的活动，建立一个多窗格UI。
- 片段可用于多种活动。
- 片段的生命周期是密切相关，其主机活动的生命周期，表示当活动暂停时，所有的片段也将停止活动。
- 片段可以实现的行为当没有用户界面组件。
- 片段加入被加入到 Android API 在Honeycomb版本的Android（API版本11）。

创建片段可以扩展**Fragment** 类并在活动的布局文件中声明片段，可以插入到活动布局的一个片段到<fragment>元素。

介绍片段之前，有一个限制，因为可以在一个特定的时间点，屏幕上只显示单个活动。所以不能够分割设备屏幕来分别控制不同部位。但随着引进片段得到了更多的灵活性，并在屏幕上同一时时间可以将一个单一的活动取消限制。现在有一个单一的acitivity， 但每个acitivity 可以包括多个片段，它们有自己的布局，活动和完整的生命周期。

下面是一个典型的例子，两个UI模块定义的片段可以组合成平板电脑的设计的一个活动，这里在手机中设计分离。



应用程序嵌入活动A中两个片段，在一个平板大小的设备上运行。然而在手机大小的屏幕上，有两个片段有足够的空间，所以Activity A包括片段物品的清单，当用户选择一篇文章时，它开始使用Activity B，包括阅读第二片段的文章。

片段生命周期

Android 的碎片有自己的生命周期，非常相似 Android 中的 Activity 。本节主要阐述其生命周期在不同阶段。

阶段 **I:** 当被创建了一个片段，它通过以下状态：

- onAttach()
- onCreate()
- onCreateView()
- onActivityCreated()

阶段 **II:** 当片段变得可见，它通过这些状态：

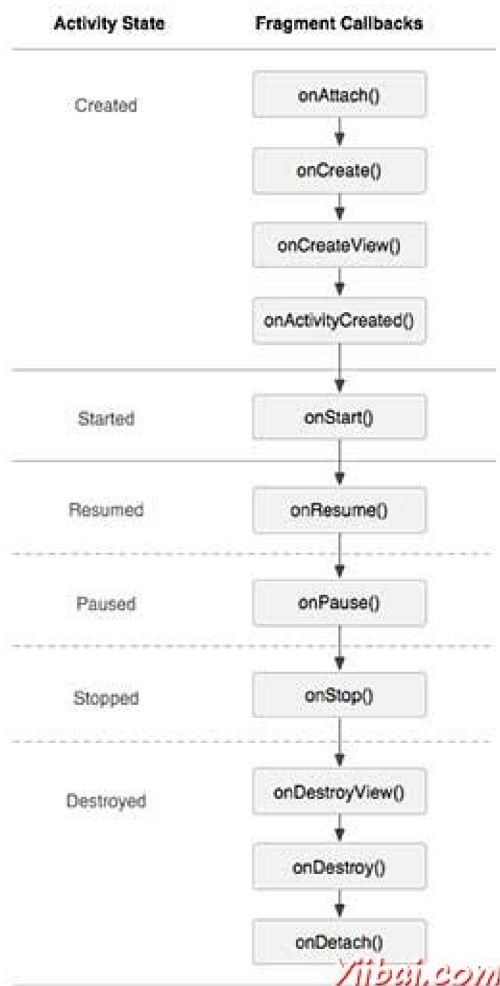
- onStart()
- onResume()

阶段 **III:** 当碎片进入后台模式，它通过这些状态：

- onPaused()
- onStop()

阶段 **IV:** 当片段被破坏，它通过以下状态：

- onPaused()
- onStop()
- onDestroyView()
- onDestroy()
- onDetach()



如何使用碎片？

这里演示简单的步骤来创建碎片：

- 首先，要决定有多少碎片要在活动中要使用。例如，要使用两个片段处理设备的横向和纵向模式。
- 在下一页的碎片数量的基础上，创建类将扩展 ***Fragment*** 类。上述片段类的回调函数。可以根据要求覆盖所有的功能。
- 对应每一个片段，需要在XML文件中创建布局。这些文件将根据布局来定义碎片。
- 最后修改活动文件替换片段，根据需要来定义实际的逻辑。

这里是重要的 覆盖在***Fragment*** 类的方法，如下列表：

- onCreate() 系统调用时创建片段。初始化片段要保留暂停或停止时的片段，然后恢复其它组成部分。
- onCreateView() 当片段第一次绘制用户界面时，系统调用这个回调。要绘制一个UI为片段，必须返回一个 View 组件，此方法是片段的根布局。返回空片段不提供一个UI。

- onPause() 系统调用此方法，作为第一次指示用户离开此片段。这通常是提交更改操作，持久化时间超过当前用户会话时间。

例子

下面的这个例子将解释如何创建片段 - *Fragments*。在这里将创建两个片段并且当其中一个使用的设备是在横向模式下，另一个片段将被用在纵向模式下。按照下面的步骤类似于在前面创建的Hello World范例：

步骤	描述
1	使用Eclipse IDE创建Android应用程序，并将其命名为MyFragments在一个包com.example.myfragments下，使用空活动。
2	主要活动文件MainActivity.java的代码修改为如下。在这里将检查设备的方向并在不同的片段之间进行切换。
3	在 com.example.myfragments包下创建两个java文件PM_Fragment.java和LM_Fragement.java来定义片段以及相关方法。
4	创建布局文件 **res/layout/lm_fragment.xml** 并布局定义这两个片段。
5	修改 **res/layout/activity_main.xml** 文件的默认内容，以包括两个片段。
6	在 **res/values/strings.xml**文件中定义所需的常量
7	运行该应用程序启动 Android 模拟器来验证应用程序所做的修改结果。

以下是主要活动文件的内容
src/com.example.mycontentprovider/MainActivity.java 修改

```
package com.example.myfragments;

import android.os.Bundle;
import android.app.Activity;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.content.res.Configuration;
import android.view.WindowManager;

public class MainActivity extends Activity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      Configuration config = getResources().getConfiguration();

      FragmentManager fragmentManager = getFragmentManager();
      FragmentTransaction fragmentTransaction =
      fragmentManager.beginTransaction();

      /**
      * Check the device orientation and act accordingly
      */
      if (config.orientation == Configuration.ORIENTATION_LANDSCAPE
         /**
         * Landscape mode of the device
         */
         LM_Fragment ls_fragment = new LM_Fragment();
         fragmentTransaction.replace(android.R.id.content, ls_fragm
      }else{
         /**
         * Portrait mode of the device
         */
         PM_Fragment pm_fragment = new PM_Fragment();
         fragmentTransaction.replace(android.R.id.content, pm_fragm
      }
      fragmentTransaction.commit();
   }

}
```

创建两个的片段文件LM_Fragement.java 和 PM_Fragment.java在 com.example.mycontentprovider 包下。

以下是LM_Fragement.java文件的内容 ：

```
package com.example.myfragments;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class LM_Fragment extends Fragment{
   @Override
   public View onCreateView(LayoutInflater inflater,
      ViewGroup container, Bundle savedInstanceState) {
      /**
       * Inflate the layout for this fragment
       */
      return inflater.inflate(
      R.layout.lm_fragment, container, false);
   }
}
```

下面是 PM_Fragement.java 文件的内容:

```
package com.example.myfragments;

import android.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class PM_Fragment extends Fragment{
   @Override
   public View onCreateView(LayoutInflater inflater,
      ViewGroup container, Bundle savedInstanceState) {
      /**
       * Inflate the layout for this fragment
       */
      return inflater.inflate(
      R.layout.pm_fragment, container, false);
   }
}
```

创建两个布局文件 **lm_fragement.xml** 和 **pm_fragment.xml** 在目录 res/layout 下。

以下是 **lm_fragement.xml** 文件的内容：

```
<?xml version="1.0" encoding="utf-8"?>
   <LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="#7bae16">

   <TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/landscape_message"
   android:textColor="#000000"
   android:textSize="20px" />

<!-- More GUI components go here  -->

</LinearLayout>
```

以下是 **pm_fragment.xml** 文件的内容：

```
<?xml version="1.0" encoding="utf-8"?>
   <LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="horizontal"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:background="#666666">

   <TextView
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/portrait_message"
   android:textColor="#000000"
   android:textSize="20px" />

<!-- More GUI components go here  -->

</LinearLayout>
```

下面 **res/layout/activity_main.xml** 文件的内容，其中包括片段：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal">

   <fragment
   android:name="com.example.fragments"
   android:id="@+id/lm_fragment"
   android:layout_weight="1"
   android:layout_width="0dp"
   android:layout_height="match_parent" />

   <fragment
   android:name="com.example.fragments"
   android:id="@+id/pm_fragment"
   android:layout_weight="2"
   android:layout_width="0dp"
   android:layout_height="match_parent" />

</LinearLayout>
```

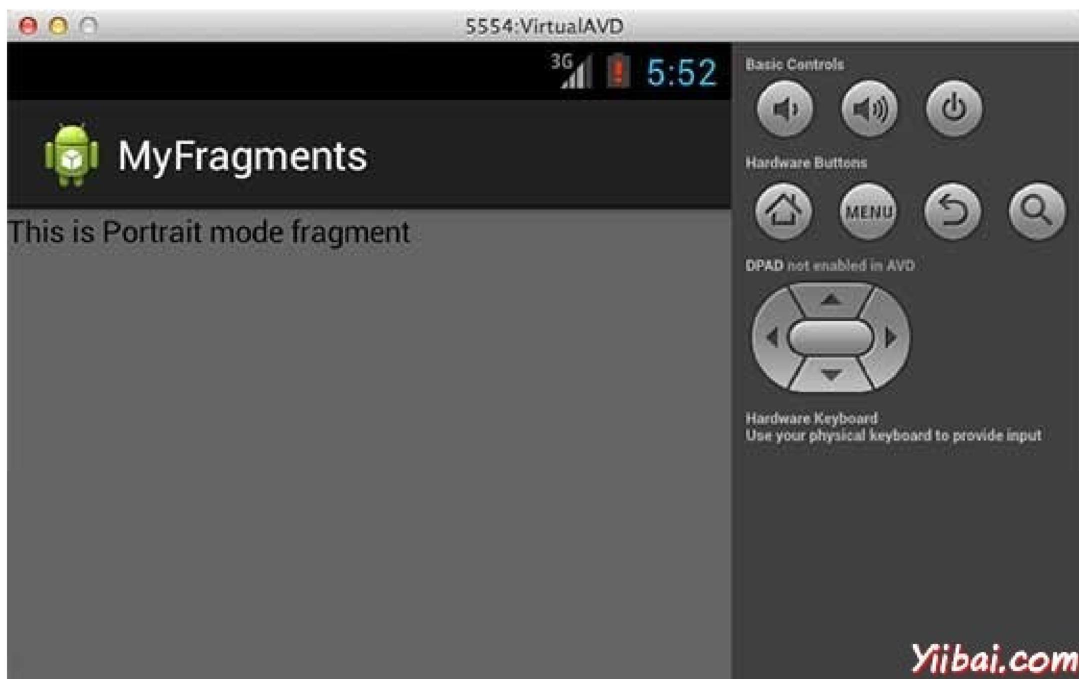
确保 **res/values/strings.xml** 文件有以下内容：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">MyFragments</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="landscape_message">This is Landscape mode fragmen
    </string>
    <string name="portrait_message">This is Portrait mode fragment
    </string>

</resources>
```

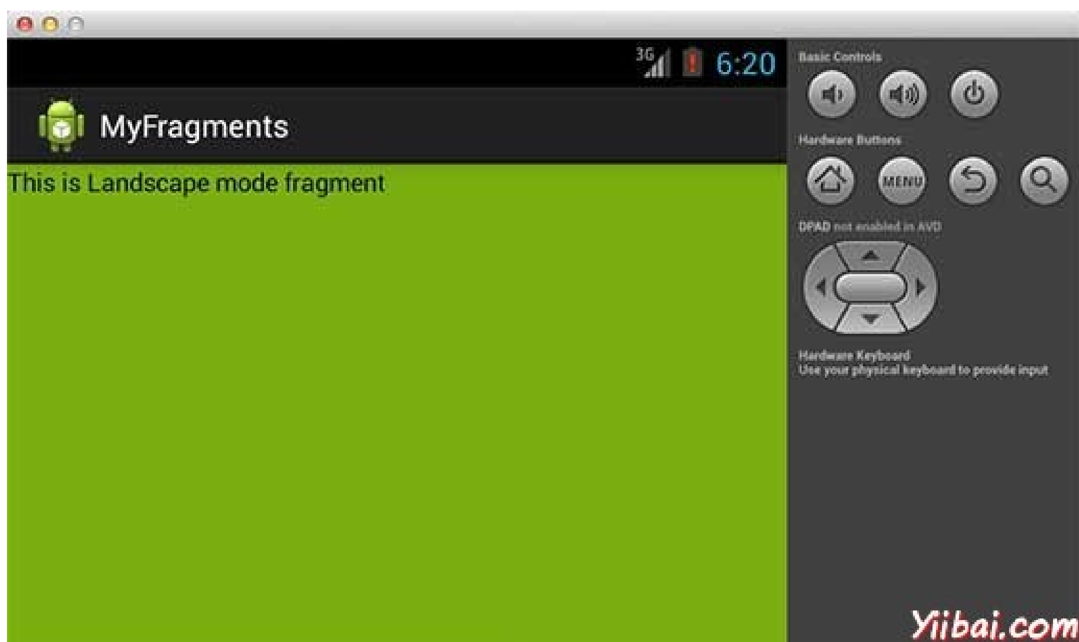
现在试着来运行 MyFragments 刚刚创建的应用程序。假设创建AVD，同时做好了环境设置。要从Eclipse运行应用程序，首先打开一个项目的活动文件，从工具栏上单击“Run”图标。 Eclipse AVD 安装应用程序，并启动它，如果设置和应用都没有问题，它会显示仿真器窗口，看到如下窗口，点击"MENU" 按钮。可能需要点耐心，因为它可能需要一段时间（易百教程提示：取决于你的电脑速度了）：



要改变模式，模拟器的屏幕，可以做以下操作：

- fn+control+F11 Mac上改变的风景，图像，反之亦然。
- ctrl+F11 在Windows.
- ctrl+F11 在 Linux.

改变了模式以后，能够看到的图形用户界面，如下已经实现了横向模式：



这样就可以使用相同的活动，但不同的GUI要通过不同的片段。根据要求可以使用不同类型的GUI组件来创建不同的GUI。

Android Intent过滤器 - Android开发教程

Android Intent 是承载一个意图，即对象。将消息从一个组件传到另一个组件，在应用程序或应用程序之外。Intent 之间沟通信息的任何应用程序的三个核心组件 - 活动，服务和广播接收器。

> 意图本身是一个Intent对象，是一种被动的数据结构保持将要执行的动作的抽象描述。

例如，让我们假设有一个Activity ，需要启动电子邮件客户端和发送电子邮件，使用Android设备。为了达到这个目的，Activity会随着适当选择器，一个ACTION_SEND发送到 Android Intent 解析器。指定的选择器提供适当的接口供用户选择如何发送电子邮件数据。

例如，有一个Activity ，需要在 Android 设备上用Web浏览器打开网址。为了达到这个目的Activity将发送ACTION_WEB_SEARCH Intent 到Android Intent 解析器，并在Web浏览器中打开给定的URL。Intent 解析器解析通过一个活动列表，并选择一个最合适的 Intent ，在这种情况下，就是Web浏览器活动。Intent通过网页解析后，网页浏览器启动 Web 浏览器活动。

分开机制对于每种类型的组件提供意图，如：活动，服务和广播接收器。

S.N.	方法& 描述
1	**Context.startActivity()** Intent对象传递给此方法来启动一个新的活动或获得现有活动来做一些新的东西。
2	**Context.startService()** Intent对象传递给此方法来启动服务或提供新的指令到一个持续的服务。
3	**Context.sendBroadcast()** Intent对象传递给这个方法消息给所有感兴趣的广播接收器。

Intent对象

Intent对象是成捆的信息，这些信息所使用的组件，它接收的意图和Android系统中的信息。

Intent对象可以基于它是什么交流或要执行，包含以下组件：

动作

强制Intent对象是一个字符串，命名要执行操作或广播意图，正在发生的动作和报告。动作在很大程度上决定意图对象的其余部分的结构如何。Intent类定义了一些动作常数对应不同的意图。下面列出的是 Android Intent标准动作

动作在一个Intent对象的 setAction() 方法可以设置，通过 getAction() 方法读取。

数据

要采取动作的数据的URI和该数据的MIME类型。例如，如果动作字段ACTION_EDIT，在数据字段将包含要显示的编辑的文档的URI。

setData()方法指定数据仅作为URI的setType()指定它只能作为一个MIME类型，和setDataAndType()指定它作为一个URI和MIME类型。读取的URI由getData()和getType()。

操作/数据对的一些例子是：

S.N.	动作/数据对和说明
1	**ACTION_VIEW content://contacts/people/1** 显示有关其标识符为“1”的人的信息。
2	**ACTION_DIAL content://contacts/people/1** 显示电话拨号程序已填充的人。
3	**ACTION_VIEW tel:123** 显示电话拨号程序已填充的给定数。
4	**ACTION_DIAL tel:123** 显示电话拨号程序已填充的给定数。
5	**ACTION_EDIT content://contacts/people/1** 编辑有关其标识符为“1”的人的信息。
6	**ACTION_VIEW content://contacts/people/** 显示人的列表，用户可以浏览。

类别

类别是 Intent 对象一个可选部分，这是一个字符串，其中包含应该处理这个 Intent 组件的附加信息。 addCategory()方法将类别添加到 Intent对象，removeCategory() 删除一个类别，GetCategories() 得到当前在对象的所有类别集合。下面是 Android意图标准类别 列表。

要查看详细的 Intent过滤器可在下一小节， 了解如何使用类别，选择合适的活动对应的意图。

附加设备

这是键-值对，可了解更多传递组件处理 intent 的信息。额外内容设置和读取使用 putExtras() 和 getExtras() 方法。这里是一个 Android intent 标准额外数据列表。

标志位

这些标志是Intent对象可选部分，并指示Android系统如何启动一个活动以及如何对待它在启动后等等。

组件名称

可选字段是一个 android 组件名称的活动，服务或BroadcastReceiver类对象。如果它没有被设置，Android使用Intent 对象的信息，找到一个合适的目标，Intent 对象传递到指定类的一个实例。

组件名称由setComponent()，setClass()设置，由 setClassName()和getComponent()读取。

Intents类型

有以下两种类型的意图支持到 Android 4.1版本

显式意图

显示意图指定目标组件的名称，通常用于应用程序内部消息 - 比如一个活动启动从服务或启动一个组活动。例如：

```
// Explicit Intent by specifying its class name
Intent i = new Intent(this, TargetActivity.class);
i.putExtra("Key1", "ABC");
i.putExtra("Key2", "123");

// Starts TargetActivity
startActivity(i);
```

隐式意图

这些意图由其的名字指定目标组件，它们通常用于应用程序内部消息 - 例如一个活动启动一个附属服务或启动一个姐妹的活动。例如：

```
// Implicit Intent by specifying a URI
Intent i = new Intent(Intent.ACTION_VIEW,
Uri.parse("http://www.example.com"));

// Starts Implicit Activity
startActivity(i);
```

目标组件接收的意图可以使用getExtras()方法来获得组件发送额外的数据源。例如：

```
// Get bundle object at appropriate place in your code
Bundle extras = getIntent().getExtras();

// Extract data using passed keys
String value1 = extras.getString("Key1");
String value2 = extras.getString("Key2");
```

例子

下面的例子展示了Android意图的功能，启动各种 Android 的内置应用。

步骤	描述
1	使用Eclipse IDE创建Android应用程序，并将其命名为IntentDemo在com.example.intentdemo包下。在创建这个项目，请确保目标SDK并编译在Android SDK的最新版本为使用更高级别的API。
2	修改 src/MainActivity.java 文件，并添加代码来定义相应的两个按钮，即两个监听器。启动浏览器，并打开电话。
3	修改布局XML文件res/layout/activity_main.xml添加三个按钮的设置为线性布局。
4	修改 res/values/strings.xml 中定义所需的常量值
5	运行该应用程序启动Android模拟器来验证应用程序所做的修改结果。

以下是修改主要活动文件 **src/com.example.intentdemo/MainActivity.java** 的内容：

```
package com.example.intentdemo;

import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      Button startBrowser = (Button) findViewById(R.id.start_browse
      startBrowser.setOnClickListener(new View.OnClickListener() {
         public void onClick(View view) {
            Intent i = new Intent(android.content.Intent.ACTION_VIE
            Uri.parse("http://www.example.com"));
            startActivity(i);
         }
      });
      Button startPhone = (Button) findViewById(R.id.start_phone);
      startPhone.setOnClickListener(new View.OnClickListener() {
         public void onClick(View view) {
            Intent i = new Intent(android.content.Intent.ACTION_VIE
            Uri.parse("tel:9510300000"));
            startActivity(i);
         }
      });
   }
   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action
      // bar if it is present.
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
   }

}
```

下面是 res/layout/activity_main.xml 文件的内容：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <Button android:id="@+id/start_browser"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/start_browser"/>

   <Button android:id="@+id/start_phone"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/start_phone" />

</LinearLayout>
```

下面 res/values/strings.xm 定义两个新的常量的内容：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">IntentDemo</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="start_browser">Start Browser</string>
    <string name="start_phone">Start Phone</string>

</resources>
```

以下是 **AndroidManifest.xml** 文件的默认内容：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/androic
    package="com.example.intentdemo"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.intentdemo.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" /

                <category android:name="android.intent.category.LAU
            </intent-filter>
        </activity>
    </application>

</manifest>
```

现在运行 IntentDemo应用程序。假设创建AVD同时做正确的环境设置。要从Eclipse运行的应用程序，打开一个项目的活动文件，并 从工具栏上单击“**Run**”图标。 Eclipse AVD安装的应用程序，并启动它，如果一切都设置和应用没有问题，它会显示以下模拟器窗口：



现在点击开始浏览按钮，这将启动一个浏览器配置并显示 http://www.example.com 如下所示：



类似的方式，可以启动手机界面使用开始电话按钮，这允许拨打已经给定的电话号码。

Intent过滤器

上面已经看到了意图如何用来调用一个活动。 Android操作系统使用过滤器来查明活动，服务和广播接收器能够处理指定的一组动作，类别。使用 <intent-filter> 元素在 manifest 文件中，列出了动作，类别和数据类型相关联的活动，服务或广播接收器。

以下是 AndroidManifest.xml 文件中的一部分，指定活动 com.example.intentdemo.CustomActivity 的两个动作，类和数据，下面是可以调用的一个例子：

```
<activity android:name=".CustomActivity"
   android:label="@string/app_name">
   <intent-filter>
      <action android:name="android.intent.action.VIEW" />
      <action android:name="com.example.intentdemo.LAUNCH" />
      <category android:name="android.intent.category.DEFAULT" />
      <data android:scheme="http" />
   </intent-filter>
</activity>
```

活动的定义是随着上面提到的过滤器，活动将使用 **android.intent.action.VIEW** 或 **com.example.intentdemo.LAUNCH** 动作提供其类别来调用这个活动，否则使用**android.intent.category.DEFAULT**。

<data> 元素活动被称为指定数据类型，上面的例子中自定义活动的数据以 "http://" 开始

有可能在只是一个意图的情况下，可以通过一个以上的活动或服务的过滤器，用户可能会被要求指定激活哪个组件。可以发现如果没有指定目标，则会引发异常。

以下测试 Android 检查在调用活动前：

- 如上图所示，但这个列表不能为空，可能会列出多个过滤器<intent-filter>动作，一个过滤器必须至少包含一个<action>元素，否则会阻止所有意图。如果有一个以上的动作列出，那么Android将尝试匹配在调用活动之前所提到的其中一个动作。
- 过滤器<intent-filter>可能列出零个，一个或一个以上。如果没有类被提到，Android也能通过这个测试，但如果超过一个类提到通过类的测试意图，在过滤器中，每一个类中的Intent对象必须符合一个类。
- 每个<data>元素可以指定一个URI和数据类型（MIME媒体类型）。单独的属性，如方案，主机，端口和路径URI的每个部分。同时包含URI和数据类型的一个Intent对象通过测试，只有当它的类型在过滤器列出的类型相匹配数据类型的一部分。

例子

下面是修改上面例子的一个例子。在这里将看到 Android 如何解决冲突，如果一个意图调用定义了两个活动，接下来如何调用自定义活动使用一个过滤器，第三个是一个在例外情况下，如果 Android 不提交适当的活动意图定义。

步骤	描述
1	使用Eclipse IDE创建 *Android* 应用程序，并将其命名为 *IntentDemo* 在一个包*com.example.intentdemo*下。在创建这个项目前，确保目标SDK编译在Android SDK的最新版本或使用更高级别的API。
2	修改 *src/MainActivity.java* 文件，并添加代码来定义相应的布局文件中定义了三个按钮的监听器。
3	添加一个新的 src/CustomActivity.java 文件，将由不同的意图来调用一个自定义的活动。
4	修改布局XML文件res/layout/activity_main.xml 添加三个按钮的线性布局。
5	添加一个布局XML文件res/layout/custom_view.xml，在其中添加一个简单的<TextView>来显示通过意图传递的数据。
6	修改 *res/values/strings.xml* 定义所需的常量值
7	修改 AndroidManifest.xml 添加 <intent-filter> 定义规则，来自定义意图调用活动。
8	运行该应用程序启动Android模拟器，并确认在应用修改变化的结果。

以下是内容是修改了主要活动文件 *src/com.example.intentdemo/MainActivity.java.*

```
package com.example.intentdemo;

import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      // First intent to use ACTION_VIEW action with correct data
      Button startBrowser_a = (Button) findViewById(R.id.start_brow
      startBrowser_a.setOnClickListener(new View.OnClickListener()
         public void onClick(View view) {
            Intent i = new Intent(android.content.Intent.ACTION_VIE
            Uri.parse("http://www.example.com"));
            startActivity(i);
```

```
         }
      });

      // Second intent to use LAUNCH action with correct data
      Button startBrowser_b = (Button) findViewById(R.id.start_brow
      startBrowser_b.setOnClickListener(new View.OnClickListener()
         public void onClick(View view) {
            Intent i = new Intent("com.example.intentdemo.LAUNCH",
            Uri.parse("http://www.example.com"));
            startActivity(i);
         }
      });

      // Third intent to use LAUNCH action with incorrect data
      Button startBrowser_c = (Button) findViewById(R.id.start_brow
      startBrowser_c.setOnClickListener(new View.OnClickListener()
         public void onClick(View view) {
            Intent i = new Intent("com.example.intentdemo.LAUNCH",
            Uri.parse("https://www.example.com"));
            startActivity(i);
         }
      });

   }

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
   // Inflate the menu; this adds items to the
   // action bar if it is present.
   getMenuInflater().inflate(R.menu.main, menu);
   return true;
   }

}
```

以下是修改的主要活动文件的内容
src/com.example.intentdemo/CustomActivity.java.

```
package com.example.intentdemo;

import android.app.Activity;
import android.net.Uri;
import android.os.Bundle;
import android.widget.TextView;

public class CustomActivity extends Activity {
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.custom_view);

      TextView label = (TextView) findViewById(R.id.show_data);

      Uri url = getIntent().getData();
      label.setText(url.toString());
   }

}
```

以下是 *res/layout/activity_main.xml* 文件的内容:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <Button android:id="@+id/start_browser_a"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/start_browser_a"/>

   <Button android:id="@+id/start_browser_b"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/start_browser_b"/>

   <Button android:id="@+id/start_browser_c"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/start_browser_c"/>

</LinearLayout>
```

下面是 res/layout/custom_view.xml 文件的内容：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"    >

   <TextView android:id="@+id/show_data"
   android:layout_width="fill_parent"
   android:layout_height="400dp"/>

</LinearLayout>
```

下面 res/values/strings.xml 文件内容中定义两个新的常量：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">IntentDemo</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="start_browser_a">Start Browser with VIEW action</
    <string name="start_browser_b">Start Browser with LAUNCH actior
    <string name="start_browser_c">Exception Condition</string>

</resources>
```

以下是 AndroidManifest.xml 文件的默认内容：

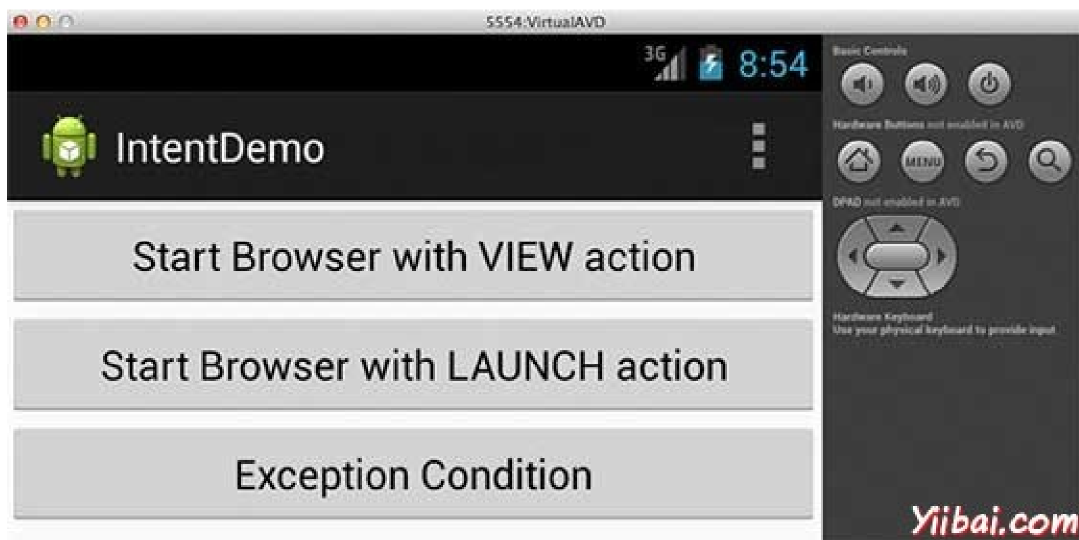
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/androic
    package="com.example.intentdemo"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.intentdemo.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" /

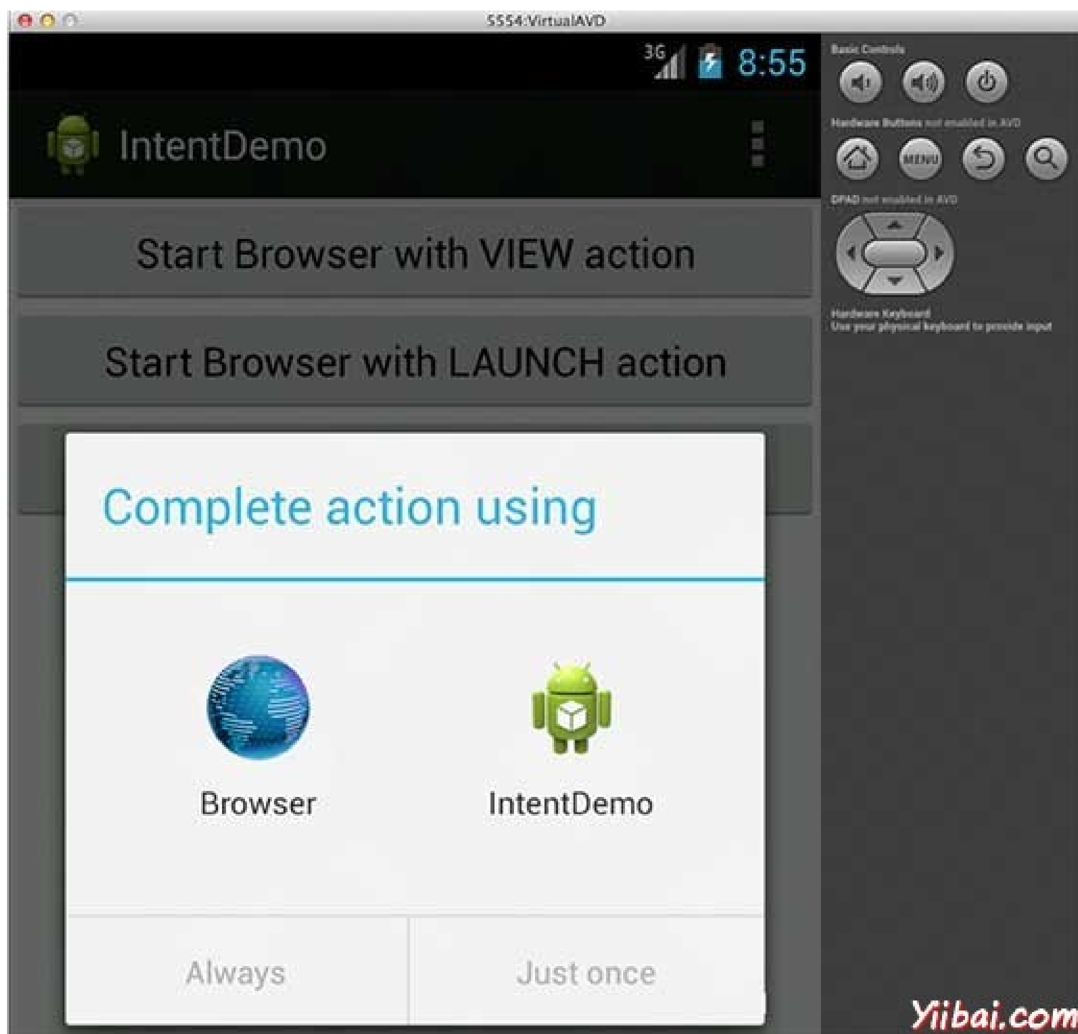
                <category android:name="android.intent.category.LAU
            </intent-filter>
        </activity>
        <activity android:name="com.example.intentdemo.CustomActivi
            android:label="@string/app_name">
           <intent-filter>
              <action android:name="android.intent.action.VIEW" />
              <action android:name="com.example.intentdemo.LAUNCH"
              <category android:name="android.intent.category.DEFAU
              <data android:scheme="http" />
           </intent-filter>
        </activity>
    </application>

</manifest>
```

要运行IntentDemo应用程序。假设创建了AVD并设置了环境。要从 Eclipse 运行应用程序，打开一个项目的活动文件，从工具栏上单击“Run” 图标。 Eclipse AVD安装的应用程序并启动它，如果一切设置和应用都没有问题，它会显示以下模拟器窗口：



现在,开始第一个按钮“Start Browser with VIEW Action”。这里定义了自定义活动过滤器“android.intent.action.VIEW”，并且已经有一个默认的活动，由Android启动Web浏览器视图定义动作，所以android 显示以下两个选项来选择要启动的活动。



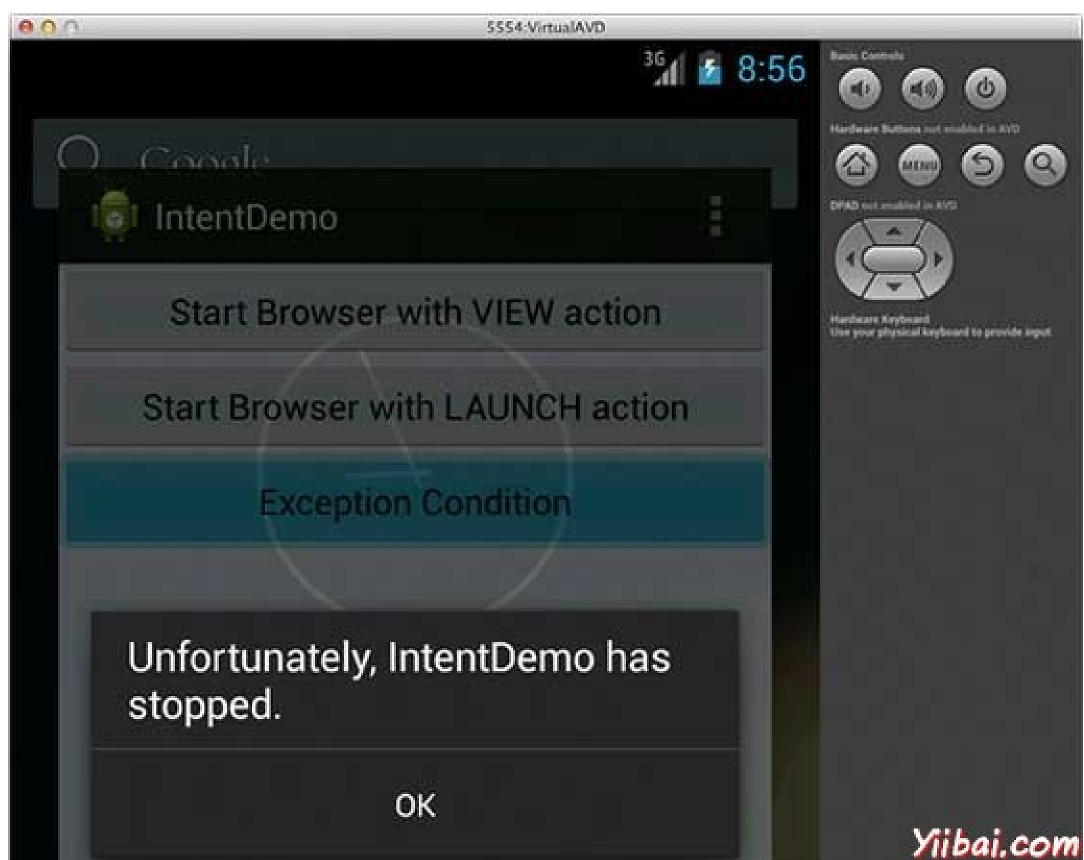
现在，如果选择浏览器，那么Android将启动网页浏览器并打开 example.com 网站，但如果选择“IndentDemo”选项，那么Android将启动CustomActivity什么也不做，只不过是捕捉传递的数据，并显示在文本视图如下：



现在使用“后退”按钮，并点击“Start Browser with LAUNCH Action”按钮，这里Android 应用过滤器来选择定义活动，只需启动自定义活动，再次显示以下画面：



同样，返回使用“后退”按钮，并点击“Exception Condition”按钮，在这里Android试图找出一个有效的过滤器，对于给定intent，它没有找到一个有效活动定义，因为在这个时候，已经数据使用HTTPS，而不是HTTP，虽然是一个正确的动作，但是Android抛出一个异常，并显示以下画面：


5554:VirtualAVD
8:56
IntentDemo
Start Browser with VIEW action
Start Browser with LAUNCH action
Exception Condition
Unfortunately, IntentDemo has stopped.
OK
Yiibai.com

Android UI布局 - Android开发教程

用户界面的基本构建块是创建View类View对象，并占据屏幕上的一个矩形区域，负责绘图和事件处理。View是用于创建交互式UI组件，它是按钮，文本框等部件的基类。

ViewGroup是View的一个子类，并提供了无形的容器，容纳其他视图或其他ViewGroup定义布局属性。

第三个层次，不同的布局是ViewGroup类的子类，一个典型的布局定义为 Android 用户界面，并且可以在运行时创建，使用 View/ViewGroup 对象可视结构或者可以声明布局，使用简单的XML文件main_layout.xml，这个文件在项目res/layout文件夹中。

本教程是更多是关于创建基于图形用户界面XML文件的布局定义。布局可以包含任何类型的部件，如按钮，标签，文本框等等。以下是一个简单的XML文件的LinearLayout 例子：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView android:id="@+id/text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="This is a TextView" />
    <Button android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="This is a Button" />

    <!-- More GUI components go here  -->

</LinearLayout>
```

布局定义之后，可以从应用程序代码加载的布局资源，在Activity.onCreate()回调实现，如下所示：

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

Android布局类型

有一些Android提供的布局，可以使用在几乎所有的Android应用程序提供不同的视图，外观和风格。

S.N.	布局和说明
1	Linear Layout LinearLayout视图组，所有的子视图在单一的方向对齐，垂直或水平。
2	Relative Layout RelativeLayout相对位置显示子视图的视图组。
3	Table Layout TableLayout一种视图，组视图分为行和列。
4	Absolute Layout AbsoluteLayout 使能够指定其子视图的确切位置。
5	Frame Layout FrameLayout 屏幕上是一个占位符，可以用它来显示一个单一的视图。
6	List View ListView显示滚动项目列表视图组。
7	Grid View GridView 网格控件是一种ViewGroup用于显示一个二维，滚动的网格的项目。

布局属性

每个布局都有一组属性，它定义布局的视觉属性。所有布局中，有几个共同的属性和其他属性布局。以下是常见的属性并可以应用到所有的布局中：

属性	描述
android:id	唯一地标识该视图的ID
android:layout_width	布局的宽度
android:layout_height	这是布局的高度
android:layout_marginTop	这是关于布局的顶侧的额外的空间
android:layout_marginBottom	在布局上的底侧的额外的空间
android:layout_marginLeft	在布局上的左侧的额外的空间
android:layout_marginRight	在右侧的布局的额外空间
android:layout_gravity	它指定子视图被定位
android:layout_weight	指定有多少布局额外的空间应该分配给视图
android:layout_x	指定布局的x坐标
android:layout_y	此指定布局的y坐标
android:layout_width	布局的宽度
android:layout_width	布局的宽度
android:paddingLeft	布局的左填充
android:paddingRight	布局的右填充
android:paddingTop	布局顶部填充
android:paddingBottom	布局底部填充

这里布局/视图的宽度和高度的尺寸可以指定在DP（密度独立像素），SP（规模独立像素），PT（点为1/72英寸），PX（像素），mm（毫米），或（英寸）。

可以指定宽度和高度精确的测量，但更多的时候，使用这些常量作为宽度或高度设置：

- android:layout_width=wrap_content 告诉视图，其内容所需要的尺寸大小本身。
- android:layout_width=fill_parent 告诉视图如其父视图一样尺寸大小。

Gravity 属性定位视图对象中起着重要的作用，它可以采取一个或多个（“|”分隔）的恒定值，具体如下：

常量	值	描述
top	0x30	推进对象到其容器的顶部，不改变其大小
bottom	0x50	推进对象到其容器的底部，不改变其大小
left	0x03	推进对象到其容器的左侧，不改变其大小
right	0x05	推进对象到其容器的右侧，不改变其大小
center_vertical	0x10	放置对象在其容器的垂直中心，在不改变其大小
fill_vertical	0x70	如果需要的话完全填满其容器增长对象的垂直尺寸
center_horizontal	0x01	放置对象在其容器的水平中心，不改变其大小
fill_horizontal	0x07	如果需要的话完全填满其容器增长的对象的水平尺寸
center	0x11	放置对象在其容器的同时在垂直和水平轴的中心，而不改变其大小
fill	0x77	如果需要的话完全填满其容器增长的对象的水平和垂直尺寸
clip_vertical	0x80	可以设置附加的选项，使顶部和/或子视图底部边缘夹在其容器的边界。剪辑将基于垂直重力：顶重力将剪辑底部边缘，一个底重力将夹的顶边，并也不会夹两侧边缘
clip_horizontal	0x08	可以设置附加的选项，使左和/或子视图右边缘夹在其容器的边界。剪辑将根据水平重力：左重力将剪辑的右边缘，右重力将剪辑的左边缘，并且也不会夹两边
start	0x00800003	推进对象到其容器的开头，不改变其大小
end	0x00800005	推进对象到其容器的末尾，不改变其大小

视图标识

一个视图对象有一个唯一的ID分配给它，用于唯一识别视图。ID在XML标签的语法是：

```
android:id="@+id/my_button"
```

以下是@+ 符号的简要说明：

- 在符号（@）开头的字符串表示XML解析器解析和扩展ID字符串的其余部分，将其识别为一个ID的资源。
- 加号（+）表示，这是一个新的资源名，必须创建并添加到资源中。要创建一个视图对象的实例，并捕捉到它的布局，使用以下命令：

```
Button myButton = (Button) findViewById(R.id.my_button);
```

Android UI控件 - Android开发教程

Android应用程序的用户界面是一切，用户可以看到并与之交互。已经解了并用它定位在活动中的各种视图的布局。本章会给详细视图的各方面。

视图(View)是一个对象绘制在屏幕上，用户可以互动的东西，ViewGroup 是一个对象，其中包含其他View（ViewGroup）的对象，并可以定义用户界面的布局。

视图可以定义在一个XML文件，它提供了一个人类可读的结构布局，类似于HTML布局。例如，一个简单的垂直布局，文本视图(TextView)和按钮(Button)看起来像这样：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/anc
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <TextView android:id="@+id/text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="I am a TextView" />
   <Button android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="I am a Button" />
</LinearLayout>
```

Android UI控件

有一些 Android 提供的UI控件，允许建立应用程序的图形用户界面。

S.N.	UI控件与说明
1	TextView 这种控制用于显示文本给用户。
2	EditText EditText是TextView预定义的子类，包括丰富的编辑功能。
3	AutoCompleteTextView AutoCompleteTextView是一个视图，它类似于EditText，不同之处是在用户键入时，它会显示自动完成建议的列表。
4	Button 按钮式可以被按压，或者点击 - 由用户执行动作。
5	ImageButton AbsoluteLayout使可以指定其子视图的确切位置。
6	CheckBox 可以由用户来切换开/关。提供一组可选择的选项并不相互排斥时候呈现用户，应该使用复选框。
7	ToggleButton 一个开/关按钮带有指示灯。
8	RadioButton 单选按钮有两种状态：选中或取消选中。
9	RadioGroup RadioGroup用于组织一个或多个单选按钮。
10	ProgressBar 进度条视图（ProgressBar）提供一些日常任务，当在后台执行任务时，给出视觉反馈。
11	Spinner 一个下拉列表，允许用户选择从一组一个值（类似HTML中的select）
12	TimePicker TimePicker视图允许用户选择一天中的时间，在24小时模式或AM/ PM模式。
13	DatePicker TimePicker视图允许用户选择一天中的时间，在24小时模式或AM/PM模式。

创建UI控件

正如在前面的章节中，视图对象可能有一个唯一的ID分配给，这个唯一识别视图树内。一个视图ID在XML标签的语法是：

```
android:id="@+id/text_id"
```

要创建一个用户界面控件/视图/小工具，必须在布局文件中定义一个视图/部件，并将其分配一个唯一的ID如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <TextView android:id="@+id/text_id"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="I am a TextView" />
</LinearLayout>
```

最后控制对象创建一个实例，并获得它的布局，使用以下命令：

```
TextView myText = (TextView) findViewById(R.id.text_id);
```

Android事件处理 - Android开发教程

事件是一种有用来收集用户与应用程序互动数据的互动组件，如按键或触摸屏等放置事件，因为每个事件从Android框架维护事件队列先入先出（FIFO）基础上的队列。可以在程序中捕获这些事件，按要求并采取适当的动作。

有以下三个概念涉及到Android事件管理：

- 事件监听器: 主要涉及建立一个Android的GUI视图类，View类提供了一些事件监听器。事件监听器是对象接收通知在事件发生时。
- 事件监听器注册: 事件注册事件处理程序被注册了事件侦听器的过程，使该处理调用用的事件侦听器触发事件。
- 事件处理程序: 当一个事件发生时，已注册的事件和事件监听器，事件监听器调用事件处理程序，这是实际处理事件的方法。

事件侦听器和事件处理程序

事件处理程序	事件监听器说明
onClick()	**OnClickListener()** 当用户点击任意或触摸或焦点事件像按钮，文字，图片等，将使用onClick()事件处理程序来处理任何部件的事件
onLongClick()	**OnLongClickListener()** 当用户点击或触摸或焦点事件像按钮，文本，图像等，为1秒以上的任何插件时被调用。使用onLongClick()事件处理程序来处理这样的事件
onFocusChange()	**OnFocusChangeListener()** 当控件失去焦点时被调用。用户进入离开视图项目。使用onFocusChange()事件处理程序来处理这样的事件
onKey()	**OnFocusChangeListener()** 当用户焦点并按下或释放装置上的硬件键时被调用。将使用onKey()事件处理程序来处理这样的事件
onTouch()	**OnTouchListener()** 当用户按下该键时及释放键，或在屏幕上的任何移动手势时被调用。使用onTouch()事件处理程序来处理这样的事件
onMenuItemClick()	**OnMenuItemClickListener()** 当用户选择一个菜单项时被调用。使用onMenuItemClick()事件处理程序来处理这样的事件

还有更多可用作为View类如：OnHoverListener，OnDragListener 等，应用程序可能需要一部分的事件侦听器。因此，建议参考官方Android应用程序开发文档，开发一个复杂的应用程序。

注册事件监听器：

事件注册事件处理程序被注册事件侦听器的过程，使处理时调用事件侦听器处理事件。虽然有一些方法注册可以任何事件的事件侦听器，但要列出只前3种方式，可以根据实际情况使用。

- 使用匿名内部类
- Activity 活动类实现Listener接口
- 使用布局文件 activity_main.xml 直接指定事件处理程序（方法）

下面将提供三种情景的详细的例子：

事件处理举例

使用匿名内部类的事件监听器注册

在这里，将创建一个匿名的执行监听，如果每个类只有一个单控制器，将参数传递给事件处理程序。在这种方法中的事件处理方法可以访问私有数据的活动。没有提及需要调用到活动。

但是，如果声明一个以上处理程序的控制器，剪切和粘贴代码的处理程序和处理程序的代码很长，代码更难维护。

以下是简单的步骤来展示我们将如何利用独立的 Listener类 注册并捕获点击（click）事件。类似的方式，可以实现所需的任何其他事件类型的侦听。

步骤	描述
1	使用Android Studio创建一个Android应用程序项目，将其命名为：EventDemo
2	修改 src/MainActivity.java 程序文件，以添加 click事件侦听器并处理程序定义的两个按钮
3	修改 *res/layout/activity_main.xml* 文件的默认内容包括Android的UI控件
4	定义*res/values/strings.xml* 文件所需的常量
5	运行该应用程序启动Android模拟器并验证应用程序所做的修改结果

以下是主 activity 文件src/com.yiibai.eventdemo/MainActivity.java 的内容。这个文件可以包括每个生命周期的根本方法。

```
package com.yiibai.eventdemo;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //--- find both the buttons---
        Button sButton = (Button) findViewById(R.id.button_s);
        Button lButton = (Button) findViewById(R.id.button_l);

        // -- register click event with first button ---
        sButton.setOnClickListener(new View.OnClickListener() {
           public void onClick(View v) {
               // --- find the text view --
               TextView txtView = (TextView) findViewById(R.id.text
               // -- change text size --
               txtView.setTextSize(14);
           }
        });

        // -- register click event with second button ---
        lButton.setOnClickListener(new View.OnClickListener() {
           public void onClick(View v) {
               // --- find the text view --
               TextView txtView = (TextView) findViewById(R.id.text
               // -- change text size --
               txtView.setTextSize(24);
           }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

}
```

下面是 ***res/layout/activity_main.xml*** 文件的内容：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

    <Button
    android:id="@+id/button_s"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:text="@string/button_small"/>

    <Button
    android:id="@+id/button_l"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:text="@string/button_large"/>

    <TextView
    android:id="@+id/text_id"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:capitalize="characters"
    android:text="@string/hello_world" />

</LinearLayout>
```

以下文件 ***res/values/strings.xml*** 定义了两个新的常量:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">EventDemo - yiibai.com</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
   <string name="button_small">小号字体</string>
   <string name="button_large">大号字体</string>

</resources>
```

以下是 ***AndroidManifest.xml*** 文件的默认内容：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/androic
    package="com.yiibai.guidemo"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.yiibai.guidemo.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" /

                <category android:name="android.intent.category.LAU
            </intent-filter>
        </activity>
    </application>

</manifest>
```

我们尝试运行EventDemo 应用程序。AVD安装的应用程序，并启动它，如果一切设置和应用都没有问题，它会显示以下模拟器窗口：



现在尝试一下，会看到两个按钮逐一的Hello World文本，字体会发生变化，对每次点击事件发生，因为注册的click事件处理方法被调用。 以上代码下载：http://pan.baidu.com/s/1hqrljx6

注册使用活动实现监听器接口

在这里，activity类实现Listener接口方法处理主活动，然后调用setOnClickListener（this）程序。

如果应用程序只有一个单一的控件这种方法是很好的，但需要做进一步的编程检查控制生成的事件（监听器类型）。第二不能将参数传递到监听器，多个控件时不能起作用。

下面是简单的步骤来展示如何实现Listener类注册并捕获click事件。类似的方式，可以实现所需的任何其他事件类型的监听。

步骤	描述
1	我们需要创建一个Android应用程序 ：EventDemo2
2	修改 src/MainActivity.java 文件的内容，以添加click事件侦听器和处理程序定义的两个按钮
3	上一个例子中的 res/layout/activity_main.xml 文件不用做任何改变，它仍将如上图所示。
4	上一个例子中的 *res/values/strings.xml* 文件不做任何变化，如上图所示。
5	运行该应用程序启动Android模拟器并验证应用程序所做的修改结果。

以下是主活动活动文件 src/com.yiibai.eventdemo2/MainActivity.java 的内容。这个文件可以包括每个生命周期基础方法。

```
package com.example.eventdemo;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends Activity implements OnClickListe

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //--- find both the buttons---
        Button sButton = (Button) findViewById(R.id.button_s);
        Button lButton = (Button) findViewById(R.id.button_l);

        // -- register click event with first button ---
        sButton.setOnClickListener(this);
        // -- register click event with second button ---
        lButton.setOnClickListener(this);
    }

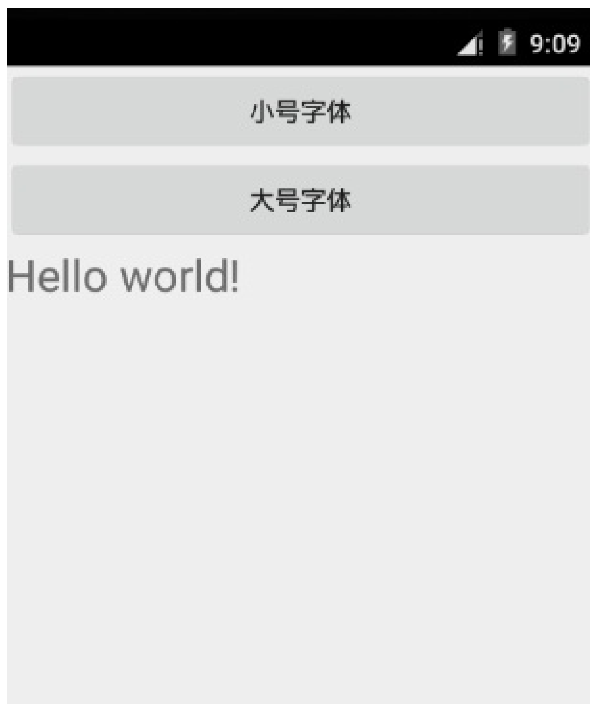
    //--- Implement the OnClickListener callback
    public void onClick(View v) {
       if(v.getId() == R.id.button_s)
       {
            // --- find the text view --
            TextView txtView = (TextView) findViewById(R.id.text_i
```

```
            // -- change text size --
            txtView.setTextSize(14);
            return;
        }
        if(v.getId() == R.id.button_l)
        {
            // --- find the text view --
            TextView txtView = (TextView) findViewById(R.id.text_ic
            // -- change text size --
            txtView.setTextSize(24);
            return;
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

}
```

我们尝试运行EventDemo2 应用程序。AVD安装的应用程序，并启动它，如果一切设置和应用都没有问题，它会显示以下模拟器窗口：



现在尝试一下，会看到两个按钮被点击后 "Hello World" 文本字体会发生变化，对每次点击事件，注册的 click 事件处理方法被调用。 上面例子程序代码下载地址：http://pan.baidu.com/s/1i3pEvpr

使用布局文件ACTIVITY_MAIN.XML注册

在这里事件处理程序Activity类没有实现监听器接口，也没有注册任何侦听器方法。相反使用布局文件（activity_main.xml），通过android:onClick属性指定的处理程序方法click事件。可以控制不同的点击事件不同的控制，通过不同的事件处理方法。

事件处理程序方法必须有一个返回类型为void，并作为一个参数来检视。方法名称可以是任意的，主类不需要实现任何特定的接口。

这种方法不会允许将参数传递给监听器，Android开发人员将很难知道哪种方法处理程序控制，需要到activity_main.xml文件查看才能知道。其次，不能处理除click事件外的任何其他事件。

以下是简单的步骤来展示如何能利用布局main.xml文件注册并捕获click事件。

步骤	描述
1	创建一个Android应用程序项目：*EventDemo3*.
2	修改*src/MainActivity.java*文件，以添加定义两个按钮的click事件侦听器和处理程序
3	修改布局文件 *res/layout/activity_main.xml*，指定这两个按钮的事件处理程序
4	文件 *res/values/strings.xml* 不用做修改，使用上面的例子中的内容就可以
5	运行该应用程序启动Android模拟器并验证应用程序所做的修改结果。

以下是修改主活动文件src/com.yiibai.eventdemo/MainActivity.java的内容。这个文件可以包括每个生命周期的基本方法。

```
package com.example.eventdemo;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends Activity{

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    //--- Implement the event handler for the first button.
    public void doSmall(View v)  {
       // --- find the text view --
       TextView txtView = (TextView) findViewById(R.id.text_id);
       // -- change text size --
       txtView.setTextSize(14);
       return;
   }
   //--- Implement the event handler for the second button.
   public void doLarge(View v)  {
       // --- find the text view --
       TextView txtView = (TextView) findViewById(R.id.text_id);
       // -- change text size --
       txtView.setTextSize(24);
       return;
   }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

}
```

将以下的 res/layout/activity_main.xml 文件的内容。在这里，我们必须给这两个按钮添加 android:onClick="methodName"， 这将注册给定的方法名，以添加单击事件处理程序。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

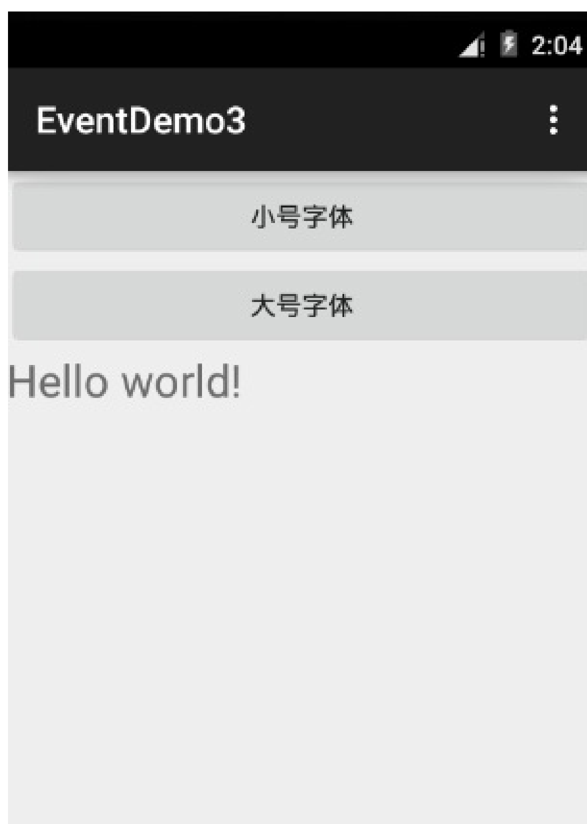
    <Button
    android:id="@+id/button_s"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:text="@string/button_small"
    android:onClick="doSmall"/>

    <Button
    android:id="@+id/button_l"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:text="@string/button_large"
    android:onClick="doLarge"/>

    <TextView
    android:id="@+id/text_id"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:capitalize="characters"
    android:text="@string/hello_world" />

</LinearLayout>
```

我们尝试运行EventDemo3 应用程序。AVD上安装的应用程序，并启动它，如果一切设置和应用都没有问题，它会显示以下模拟器窗口：



现在尝试一下，会看到两个按钮的 Hello World文本的字体会发生变化，对每次点击事件，注册的click事件处理方法被调用。 以上代码下载地下：http://pan.baidu.com/s/1nthyBDR

Android样式和主题 - Android开发教程

如果已经熟悉在网页设计中的层叠样式表（CSS），了解Android样式也是非常相似。每个 Android 窗口小部件，可以设置更改应用程序外观风格相关的属性。样式可以指定属性，如高度，填充，字体颜色，字体大小，背景颜色以及其它。

可以指定这些属性在布局文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <TextView
   android:id="@+id/text_id"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:capitalize="characters"
   android:textColor="#00FF00"
   android:typeface="monospace"
   android:text="@string/hello_world" />

</LinearLayout>
```

不过这样一来，我们需要定义的样式属性，每个属性分别用于源代码维护的角度来看这是非常不好的。因此，样式定义应该放在单独的文件，如下解释。

定义样式

样式可以定义在一个单独的XML指定布局的XML资源文件。此XML文件位于res/values/ 项目目录，强制性的样式文件中<resources>作为根节点，XML文件名称是任意，但它必须使用.xml扩展名。

可以定义每个文件中使用的多种样式<style>标签，但要使用唯一的名称来标识此样式。 Android 样式属性设置使用的<item>标签，如下图所示：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <style name="CustomFontStyle">
      <item name="android:layout_width">fill_parent</item>
      <item name="android:layout_height">wrap_content</item>
      <item name="android:capitalize">characters</item>
      <item name="android:typeface">monospace</item>
      <item name="android:textSize">12pt</item>
      <item name="android:textColor">#00FF00</item>/>
   </style>
</resources>
```

> 这里<item>里边的值可以是一个关键字串，十六进制的颜色，参考到另一个资源类型，或其他的值取决于样式属性。

使用样式

样式定义之后，就可以用它在XML布局文件使用样式属性，如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <TextView
   android:id="@+id/text_id"
   style="@style/CustomFontStyle"
   android:text="@string/hello_world" />

</LinearLayout>
```

要理解这个概念涉及到Android 的样式，可以检查 样式实例。

样式继承

Android支持级联样式表在网页设计风格非常类似继承这种方式。可以使用这个继承现有的样式属性，然后定义想要更改或添加属性。

其操作简单，创建一个新的的样式继承LargeFont上述CustomFontStyle风格定义，但字体的大小变大，可以编写这样的新的样式：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <style name="CustomFontStyle.LargeFont">
      <item name="android:textSize">20ps</item>
   </style>
</resources>
```

@style/CustomFontStyle.LargeFont 的XML布局文件，可以参考这个新的样式。可以继续秉承这样多次，只要愿意，周期通过链接名称。例如，可以扩展FontStyle.LargeFont的是红色的，如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <style name="CustomFontStyle.LargeFont.Red">
      <item name="android:textColor">#FF0000</item>/>
   </style>
</resources>
```

继承这种技术通过链接在一起的名字仅适用于自己的资源定义的样式。不能继承：Android内置样式的这种方式。要引用一个Android内置风格，如TextAppearance，必须使用父属性，如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <style name="CustomFontStyle" parent="@android:style/TextAppeara
      <item name="android:layout_width">fill_parent</item>
      <item name="android:layout_height">wrap_content</item>
      <item name="android:capitalize">characters</item>
      <item name="android:typeface">monospace</item>
      <item name="android:textSize">12pt</item>
      <item name="android:textColor">#00FF00</item>/>
   </style>
</resources>
```

Android 主题

希望能够理解样式的概念，现在让我们去了解什么是主题。主题是什么，主题只不过是要Android应用到整个活动或应用程序中统一样式，而不是一个单独的视图样式。

因此，当一个样式应用为主题，将适用于每一个活动或应用程序视图它支持每个样式属性。例如，可以应用一个主题Activity 活动的的同一CustomFontStyle风格，然后内部的所有文本，活动都会有绿色环保等宽字体。

要设置应用程序的所有活动的主题，打开AndroidManifest.xml文件，编辑<application>标签包含了android:theme 属性的风格名称。例如：

```
<application android:theme="@style/CustomFontStyle">
```

但是，如果想有一个主题，只是在应用程序的一个Activity 活动，然后添加android:theme属性到<activity>标签。例如：

```
<activity android:theme="@style/CustomFontStyle">
```

有一些由Android定义的默认主题，可以直接使用或继承父属性如下：

```
<style name="CustomTheme" parent="android:Theme.Light">
    ...
</style>
```

要理解这个概念，有关Android的主题，可以查看 主题示例。

默认样式和主题

Android平台提供了一个大集合，可以在应用程序中使用的风格和主题。可以在R.style类的参考找到所有可用的样式。要使用这里列出的风格，在一个样式名替换所有下划线。例如，可以应用Theme_NoTitleBar主题使用如“@android:style/Theme.NoTitleBar”。可以看到下面的源代码为Android的风格和主题：

- Android Styles (styles.xml)
- Android Themes (themes.xml)

Android自定义组件 - Android开发教程

Android提供了一个预建的部件，如Button, TextView, EditText, ListView, CheckBox, RadioButton, Gallery, Spinner, AutoCompleteTextView等可以直接使用在Android应用程序开发中，但有可能还有一种情况，当不满意现有可用的窗口小部件的功能。 Android 提供创建自定义组件功能，定制以满足需求。

如果只需要进行小的调现有的小工具或布局，可以简单的子类的小工具或布局和覆盖它的方法，这将精确地控制屏幕元素的外观和功能。

本教程介绍了如何创建自定义视图，并利用它们在应用程序，如下步骤。

创建一个简单的自定义组件

最简单的创建自定义的组件方法是扩展现有的widget类或子类，如果想扩展现有部件，如Button, TextView, EditText, ListView, CheckBox等，否则可以从android.view.View类开始扩展。

在其最简单的形式，编写构造函数对应的所有基类的构造函数。例如，如果要扩展TextView 创建DateView 以下三个构造，创建DateView类 ：

```
public class DateView extends TextView {
   public DateView(Context context) {
      super(context);
      //--- Additional custom code --
   }

   public DateView(Context context, AttributeSet attrs) {
      super(context, attrs);
      //--- Additional custom code --
   }

   public DateView(Context context, AttributeSet attrs, int defStyl
      super(context, attrs, defStyle);
      //--- Additional custom code --
   }
}
```

TextView 的子类DateView已经创建，所以可以获得有关TextView 的所有属性、方法和事件，能够使用不需要任何进一步的实现。这里将实现额外的自定义功能在自己编写的代码，如下面的例子解释。

如果要求执行自定义绘图/定制部件的尺寸，那么需要重写 onMeasure(int widthMeasureSpec, int heightMeasureSpec) 和 onDraw(Canvas canvas) 方法。如果不打算调整或变更内置组件的形状，那么并不需要使用这些方法在自定义组件。

布局管理报告部件的宽度和高度需要协调 onMeasure() 方法，需要调用 setMeasuredDimension(int width, int height)，这种方法来报告尺寸大小。

可以执行自定义绘图里Canvas 的onDraw(Canvas canvas) 方法，其中 android.graphis.Canvas其对应 Swing 是非常相似的，drawRect(), drawLine(), drawString(), drawBitmap() 等，可以用它来绘制组件。

完成了一个自定义组件的实现之后，通过扩大现有的部件，将能够实例化这些自定义组件在应用程序开发两种方式：

Activity类实例内使用代码

这是非常相似的方式实例化自定义组件实例的方式，在活动类的内置部件。例如，可以使用下面的代码实例上面定义的自定义组件：

```
@Override
 protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);

     DateView dateView = new DateView(this);
     setContentView(dateView);
 }
```

查看这个例子来了解如何使用代码里面活动实例化一个基本的Android自定义组件。

使用布局XML文件实例

使用传统布局XML文件实例的内置部件，相同的概念适用于自定义部件，因此将能够实例化自定义组件布局XML文件，解释如下。在com.yiibai.dateviewdemo包，已经把所有的代码相关DateView 和 DateView 类，已经把自定义组件的完整的逻辑的Java类名。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <com.yiibai.dateviewdemo.DateView
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:textColor="#fff"
     android:textSize="40sp"
     android:background="#000"
     />
</RelativeLayout>
```

要注意，在这里我们使用的所有 TextView 属性以及自定义组件没有任何变化。类似的方式能够使用所有的事件、方法，以及DateView组件。

通过这个例子，了解如何使用布局XML文件实例化一个基本的Android自定义组件。

使用自定义属性的自定义组件

我们已经看到可以如何扩展功能的内置部件，但上面给出两个例子中看到，扩展组件，可以利用它的父类的所有默认属性。但考虑到一种情况，当想从头开始创建自己的属性。下面是一个简单的程序创建和使用Android的自定义组件的新属性。这里介绍三个属性，并使用它们，如下所示：

```
<com.yiibai.dateviewdemo.DateView
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:textColor="#fff"
   android:textSize="40sp"
   custom:delimiter="-"
   custom:fancyText="true"
/>
```

第1步

第一步，使用自定义的属性在 res/values/ 目录下创建新XML文件中定义attrs.xml。看看一个例子文件 attrs.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <declare-styleable name="DateView">
   <attr name="delimiter" format="string"/>
   <attr name="fancyText" format="boolean"/>
   </declare-styleable>
</resources>
```

这里 name=value 就是要使用的布局XML文件中并作为属性，format=type 属性的类型。

第2步

第二个步骤将是从布局XML文件中读取这些属性，并将其设置为组件。这个逻辑将获得通过属性集的构造函数，因为这是包含XML属性。要读取XML中的值，首先需要从AttributeSet创建一个TypedArray，然后用它来读取和设置值，如下面的示例代码所示：

```
TypedArray a = context.obtainStyledAttributes(attrs, R.styleable.Da

final int N = a.getIndexCount();
for (int i = 0; i < N; ++i)
{
   int attr = a.getIndex(i);
   switch (attr)
   {
      case R.styleable.DateView_delimiter:
         String delimiter = a.getString(attr);
         //...do something with delimiter...
         break;
      case R.styleable.DateView_fancyText:
         boolean fancyText = a.getBoolean(attr, false);
         //...do something with fancyText...
         break;
   }
}
a.recycle();
```

第3步

最后，可以使用布局XML文件中定义的属性如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:custom="http://schemas.android.com/apk/res/com.yiibai.dat
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <com.yiibai.dateviewdemo.DateView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColor="#fff"
    android:textSize="40sp"
    custom:delimiter="-"
    custom:fancyText="true"
    />

</RelativeLayout>
```

重要的部分是 xmlns:custom="http://schemas.android.com/apk/res/com.yiibai.dateviewdemo"。需要注意的是http://schemas.android.com/apk/res/将保持原样，但最后一部分需要设置包名，也可以使用任何xmlns：在这个例子中，使用的是custom，但可以使用任何喜欢的名字。

看看这个例子，以了解如何创建自定义属性Android自定义组件 的简单步骤。

Android拖放 - Android开发教程

Android 拖/放框架允许用户将数据从一个View到另一个视图在当前布局中使用图形化的拖放动作。该框架包括以下三个重要组成部分，支持拖放功能：

- 拖动事件类
- 拖动监听器
- 辅助方法和类

拖放过程

基本上有四个步骤，在拖放过程或状态：

- 开始：此事件发生时开始拖动布局中的一个项目，应用类的tartDrag()方法来告诉系统启动拖动。startDrag()方法的参数提供被拖动的数据，这些是数据的元数据和一个回调用于绘制的拖动阴影。

  该系统首先通过回调应用程序，以获得一个拖阴影的响应。然后显示在设备上的拖影。

  接下来，系统会发送拖曳事件动作类型ACTION_DRAG_STARTED在当前布局中的所有视图对象拖动事件监听器注册。

  若要继续接收拖动事件，其中包括一个可能的放事件，一个拖事件侦听器必须返回true，如果拖动事件侦听器返回false，那么它不会收到拖动事件当前的操作，直到系统发送一个动作类型的拖曳事件ACTION_DRAG_ENDED。

- 继续：用户可以继续拖动。系统发送ACTION_DRAG_ENTERED动作，其次注册拖动事件侦听器的视图中拖动点进入ACTION_DRAG_LOCATION动作。响应该事件监听器可以选择改变其对象的外观或可以凸显其观点反应。用户移动拖动视图的边框阴影外拖曳事件侦听器接收ACTION_DRAG_EXITED动作。
- 拖动：用户释放拖动的项目视图的边框内。该系统发送对象的侦听器拖动事件使用动作类型ACTION_DROP。
- 结果：操作类型ACTION_DROP之后，系统发出一个操作类型ACTION_DRAG_ENDED表示拖动操作结束的拖动事件。

DragEvent 类

dragEvent代表一个事件，就会由系统送出拖放操作过程中在不同的时间。这个类提供了一些常量和重要的方法在使用拖/放过程。

常量

以下是所有常量作为部分 dragEvent 类整数。

S.N.	常量说明
1	**ACTION_DRAG_STARTED** 拖放操作的开始的信号
2	**ACTION_DRAG_ENTERED** 一种视图拖动点已进入视图的边框的信号
3	**ACTION_DRAG_LOCATION** 发送到ACTION_DRAG_ENTERED后的视图，如果拖影依然是查看对象的边框内
4	**ACTION_DRAG_EXITED** 信号的用户移动拖动阴影视图的边框之外
5	**ACTION_DROP** 信号到View用户发布了拖影，而阻力点就是视图的边框内
6	**ACTION_DRAG_ENDED** 视图拖放操作已经结束信号

方法

以下是作为部分的 dragEvent 类提供一些重要的和最常用的方法。

S.N.	常量说明
1	**int getAction()** 检查此事件的动作值
2	**ClipData getClipData()** 返回对象到系统调用ClipData()发送作为到startDrag一部分部分
3	**ClipDescription getClipDescription()** 返回包含在ClipData的ClipDescription对象
4	**boolean getResult()** 返回拖放操作的结果的指示
5	**float getX()** 获取阻力的X坐标点
6	**float getY()** 获取阻力的Y坐标点
7	**String toString()** 返回DragEvent对象的字符串表示

监听拖放事件

如果想要的任何布局内视图响应拖动事件，那么视图要么实现 View.OnDragListener 或者设置 onDragEvent(DragEvent) 回调方法。当系统调用的方法或监听器，它传递给上述 dragEvent 对象。可以查看对象的监听器和一个回调方法。如果发生这种情况，系统首先调用监听器，然后定义回调监听器返回 true。

组合 onDragEvent(DragEvent)方法 和 View.OnDragListener， 类似于 onTouchEvent() 和 View.OnTouchListener 使用在旧版本 Android 触摸事件的组合。

开始拖动事件

开始创建ClipData和移动数据ClipData.Item。作为ClipDataobject 的一部分提供的元数据被存储在ClipDescription内ClipData对象。对于拖放操作，并不代表数据移动，可能想使用空（null）而不是实际的对象。

下一步，可以扩展 View.DragShadowBuilder 创建一个拖动视图，或者使用View.DragShadowBuilder(View) 创建一个默认的大小相同的View参数传递给它的拖影，触摸拖动阴影点集中在拖影。

示例

下面的例子显示了一个简单的拖放示例中使用View.setOnLongClickListener() 事件侦听器和 View.OnDragEventListener().函数。

步骤	描述
1	使用Android Studio创建Android应用程序，并将它命名为：DragNDropDemo。在创建这个项目，确保目标SDK和编译在Android SDK的最新版本或使用更高级别的API。
2	修改 *src/MainActivity.java* 文件，并添加定义事件侦听器的代码，以及一个回调方法，在这个例子中使用Logo图像
3	复制图片logo.png到*res/drawable-** 文件夹。可以使用的情况下，要为他们提供了不同的设备有不同的分辨率的图像
4	修改布局文件 _res/layout/activity_main.xml _l定义logo图片的默认视图
5	运行该应用程序启动 Android模拟器并验证应用程序所做的修改结果。

以下是修改主活动文件 ***src/com.yiibai.dragndropdemo/MainActivity.java*** 。这个文件可以包括每个生命周期基本方法。

```
package com.yiibai.dragndropdemo;

import android.os.Bundle;
import android.app.Activity;
import android.content.ClipData;
import android.content.ClipDescription;
import android.util.Log;
import android.view.DragEvent;
import android.view.View;
import android.view.View.DragShadowBuilder;
import android.view.View.OnDragListener;
import android.widget.*;

public class MainActivity extends Activity{
```

```
    ImageView ima;
    private static final String IMAGEVIEW_TAG = "Android Logo";
    String msg;

    private android.widget.RelativeLayout.LayoutParams layoutParams;

    @Override
    public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);

       ima = (ImageView)findViewById(R.id.iv_logo);
       // Sets the tag
       ima.setTag(IMAGEVIEW_TAG);

       ima.setOnLongClickListener(new View.OnLongClickListener() {
          @Override
          public boolean onLongClick(View v) {
             ClipData.Item item = new ClipData.Item((CharSequence)v.

             String[] mimeTypes = {ClipDescription.MIMETYPE_TEXT_PLA
             ClipData dragData = new ClipData(v.getTag().toString(),
             mimeTypes, item);

             // Instantiates the drag shadow builder.
             View.DragShadowBuilder myShadow = new DragShadowBuilder

             // Starts the drag
             v.startDrag(dragData,  // the data to be dragged
             myShadow,  // the drag shadow builder
             null,      // no need to use local data
             0          // flags (not currently used, set to 0)
             );
             return true;
          }
       });

       // Create and set the drag event listener for the View
       ima.setOnDragListener( new OnDragListener(){
          @Override
          public boolean onDrag(View v,  DragEvent event){
          RelativeLayout.LayoutParams layoutParams = (RelativeLayout
          switch(event.getAction())
          {
             case DragEvent.ACTION_DRAG_STARTED:
                layoutParams = (RelativeLayout.LayoutParams)
                v.getLayoutParams();
                Log.d(msg, "Action is DragEvent.ACTION_DRAG_STARTED"
                // Do nothing
                break;
             case DragEvent.ACTION_DRAG_ENTERED:
                Log.d(msg, "Action is DragEvent.ACTION_DRAG_ENTERED"
                int x_cord = (int) event.getX();
```

```
                int y_cord = (int) event.getY();
                break;
             case DragEvent.ACTION_DRAG_EXITED :
                Log.d(msg, "Action is DragEvent.ACTION_DRAG_EXITED")
                x_cord = (int) event.getX();
                y_cord = (int) event.getY();
                layoutParams.leftMargin = x_cord;
                layoutParams.topMargin = y_cord;
                v.setLayoutParams(layoutParams);
                break;
             case DragEvent.ACTION_DRAG_LOCATION  :
                Log.d(msg, "Action is DragEvent.ACTION_DRAG_LOCATION
                x_cord = (int) event.getX();
                y_cord = (int) event.getY();
                break;
             case DragEvent.ACTION_DRAG_ENDED   :
                Log.d(msg, "Action is DragEvent.ACTION_DRAG_ENDED")
                // Do nothing
                break;
             case DragEvent.ACTION_DROP:
                Log.d(msg, "ACTION_DROP event");
                // Do nothing
                break;
             default: break;
             }
             return true;
          }
       });
    }
}
```

下面是 **res/layout/activity_main.xml** 文件的内容：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/a
    android:id="@+id/container"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/iv_logo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/logo"
        android:contentDescription="@string/drag_drop"  />

</RelativeLayout>
```

下面文件 res/values/strings.xml 的内容中定义两个新的常量：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">DragNDropDemo</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="drag_drop">Click on the image to drag and drop</s

</resources>
```

以下是 **AndroidManifest.xml** 文件的默认内容：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/androic
    package="com.yiibai.guidemo"
    android:versionCode="1"
    android:versionName="1.0" >

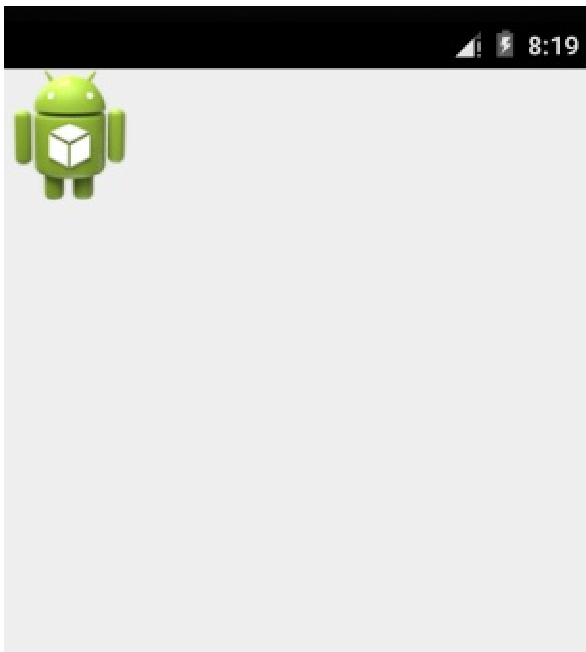
    <uses-sdk
        android:minSdkVersion="16"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.yiibai.guidemo.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" /

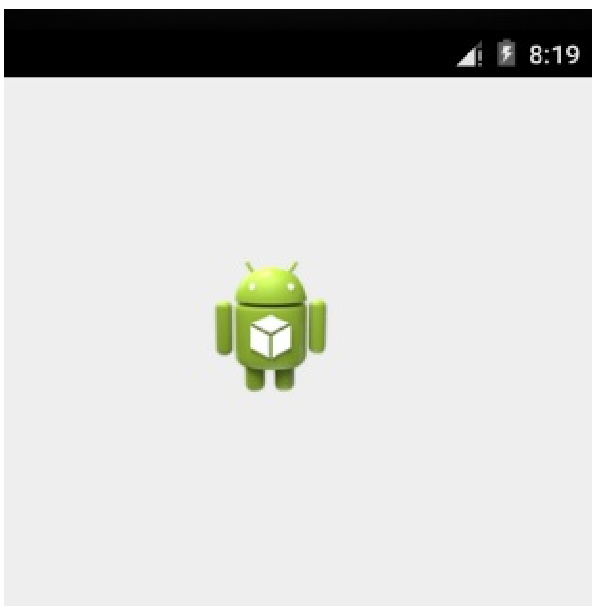
                <category android:name="android.intent.category.LAU
            </intent-filter>
        </activity>
    </application>

</manifest>
```

我们尝试运行 DragNDropDemo 应用程序。AVD安装的应用程序，并启动它，如果一切设置和应用都没有问题，它会显示以下模拟器窗口：



现在长时间点击显示Android的标志，会看到标志图像经过1秒长的点击它，开始拖动图像的时候移动了一点。可以拖动它在屏幕上，并把它放在一个新的位置。



以下代码下载：http://pan.baidu.com/s/1eQIQIjw

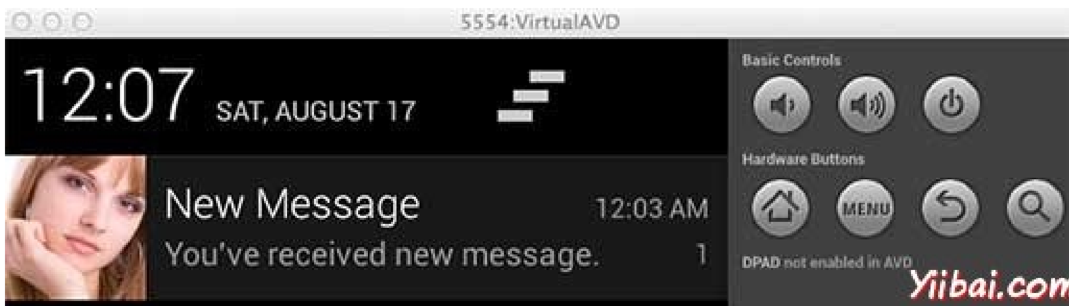
Android通知 - Android开发教程

Android的Toast 类提供了一个方便的方式来显示用户的警告信息，但这些警告不是持久性的，这意味着警告闪烁在屏幕上几秒钟后就消失了。

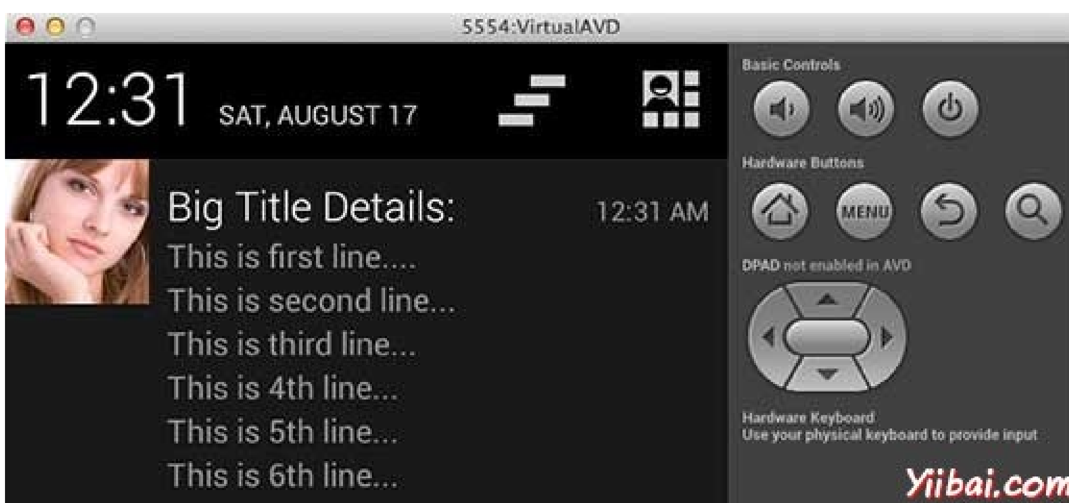
对于特别重要的要提供给用户的消息，需要有更持久性的方法。 Anotification是一种消息可以显示在设备的顶部的通知栏或状态栏中。



要看到通知的细节，选择图标显示通知抽屉里有详细的有关通知。模拟器虚拟设备工作，按一下向下拖动状态栏将它展开，将显示详细信息如下。这将是64 sp高的普通视图。



上述扩大的形式可以放到一个大的视图，有关通知的更多细节。可以添加最多六行的通知。下面的截图显示了这样的通知。



创建和发送通知

使用简单的方法来创建一个通知。按照以下步骤在应用程序创建一个通知：

第1步 - 创建通知生成器

作为第一步创建一个通知构造器，使用NotificationCompat.Builder.build()。使用通知Builder来设置属性，如各种通知其小型和大型图标，标题，优先级等。

```
NotificationCompat.Builder mBuilder = new NotificationCompat.Builde
```

第2步 - 设置通知属性

在创建Builder对象之后，可以按要求使用生成器创建通知对象。这是强制性的，以至少下列设置：

- 一个小图标，由 setSmallIcon() 设置
- 一个标题，由setContentTitle() 设置
- 详细内容由 setContentText() 设置

```
mBuilder.setSmallIcon(R.drawable.notification_icon);
mBuilder.setContentTitle("Notification Alert, Click Me!");
mBuilder.setContentText("Hi, This is Android Notification Detail!")
```

通知有很多可选的属性，可以设置。要更多地了解它们，请参考 NotificationCompat.Builder 文档。

第3步 - 动作附加

这是一个可选的部分，并要求如果要附加一个动作的通知。动作可以让用户直接从通知到应用程序中的活动，在那里它们可以在一个或多个事件，或做进一步的工作。

动作定义通过PendingIntent 在应用程序中的活动意图。要关联PendingIntent 手势请调用适当NotificationCompat.Builder 方法。例如，如果想开始活动，当用户点击通知文本通知抽屉 PendingIntent 调用setContentIntent()。

PendingIntent对象表示应用程序的执行一个动作，在以后的时间里查看应用程序是否正在运行。

堆栈builder对象将包含一个人工后退堆栈活动。确保向后导航的活动在应用程序的主屏幕。

```
Intent resultIntent = new Intent(this, ResultActivity.class);
TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
stackBuilder.addParentStack(ResultActivity.class);

// Adds the Intent that starts the Activity to the top of the stack
stackBuilder.addNextIntent(resultIntent);
PendingIntent resultPendingIntent =
        stackBuilder.getPendingIntent(
            0,
            PendingIntent.FLAG_UPDATE_CURRENT
        );
mBuilder.setContentIntent(resultPendingIntent);
```

第4步 - 发出通知

最后，调用NotificationManager.notify() 发送通知，通知对象传递到系统。通知之前，确保调用NotificationCompat.Builder.build()方法生成器对象。这种方法结合了所有的选择，设置并返回一个新的Notificationobject。

```
NotificationManager mNotificationManager =
    (NotificationManager) getSystemService(Context.NOTIFICATION_SER

// notificationID allows you to update the notification later on.
mNotificationManager.notify(notificationID, mBuilder.build());
```

NotificationCompat.Builder类

NotificationCompat.Builder类可以更容易控制标志，以及帮助构建典型通知布局。以下是 NotificationCompat.Builder类的一些重要的和最常用的方法的一部分。

S.N.	常量& 描述
1	**Notification build()** 结合所有已设置的选项，并返回一个新的 Notification 对象
2	**NotificationCompat.Builder setAutoCancel (boolean autoCancel)** 设置此标志将使它以便当用户点击它在面板中的通知被自动取消
3	**NotificationCompat.Builder setContent (RemoteViews views)** 提供定制RemoteViews使用来代替标准之一
4	**NotificationCompat.Builder setContentInfo (CharSequence info)** 设置大文本的通知的右侧
5	**NotificationCompat.Builder setContentIntent (PendingIntent intent)** 提供一个PendingIntent通知被点击时发出
6	**NotificationCompat.Builder setContentText (CharSequence text)** 设置通知的文本（第二行），在一个标准的通知
7	**NotificationCompat.Builder setContentTitle (CharSequence title)** 设置通知的文本（第一行），在一个标准的通知
8	**NotificationCompat.Builder setDefaults (int defaults)** 设置将要使用的默认通知选项
9	**NotificationCompat.Builder setLargeIcon (Bitmap icon)** 设置显示在自动收报机和通知大图标
10	**NotificationCompat.Builder setNumber (int number)** 在通知的右侧设置大的数字
11	**NotificationCompat.Builder setOngoing (boolean ongoing)** 设置这是否是一个持续的通知
12	**NotificationCompat.Builder setSmallIcon (int icon)** 设置小图标在通知使用布局
13	**NotificationCompat.Builder setStyle (NotificationCompat.Style style)** 在构建时应用添加丰富的通知样式
14	**NotificationCompat.Builder setTicker (CharSequence tickerText)** 设置在第一个通知到达时显示在状态栏中的文本
15	**NotificationCompat.Builder setVibrate (long[] pattern)** 设置振动模式的使用
16	**NotificationCompat.Builder setWhen (long when)** 设置该事件发生的时间。在面板的通知是由这个时间进行排序

示例

以下示例显示 Android 的通知功能，NotificationCompat.Builder类已在Android4.1中引入。

步骤	描述
1	使用Android Studio创建一个Android应用程序，并将它命名为：NotificationDemounder。在创建这个项目时确保目标SDK和编译在Android SDK的最新版本或更高级别的API。
2	修改 *src/MainActivity.java* 文件，并添加定义三种方法startNotification(), cancelNotification()和updateNotification()，以涵盖与Android的通知的最大功能的代码。
3	创建一个新的src/NotificationView.java，这将被用于显示新的布局作为新的活动将被启动的一部分，当用户将点击通知
4	复制图片woman.png在RES/ drawable-*文件夹，这个图片将被用作通知图标。可以使用的情况下，要为他们提供了不同的设备有不同的分辨率的图片
5	修改布局XML文件 *res/layout/activity_main.xml* 添加三个按钮的线性布局
6	创建一个新的布局XML文件 *res/layout/notification.xml*。这将被用来作为布局文件为新的活动，将启动时用户将点击任何通知
7	修改 *res/values/strings.xml* 中定义所需的恒定值
8	运行该应用程序时启动Android模拟器并验证应用程序所做的修改结果

以下是修改主要活动文件src/com.yiibai.notificationdemo/MainActivity.java 的内容。这个文件可以包括每个生命周期基本方法。

```
package com.example.notificationdemo;

import android.os.Bundle;
import android.app.Activity;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.TaskStackBuilder;
import android.content.Context;
import android.content.Intent;
import android.support.v4.app.NotificationCompat;
import android.util.Log;
import android.view.View;
import android.widget.Button;

public class MainActivity extends Activity {
   private NotificationManager mNotificationManager;
   private int notificationID = 100;
   private int numMessages = 0;
```

```
    protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);

       Button startBtn = (Button) findViewById(R.id.start);
       startBtn.setOnClickListener(new View.OnClickListener() {
          public void onClick(View view) {
             displayNotification();
          }
       });

       Button cancelBtn = (Button) findViewById(R.id.cancel);
       cancelBtn.setOnClickListener(new View.OnClickListener() {
          public void onClick(View view) {
             cancelNotification();
          }
       });

       Button updateBtn = (Button) findViewById(R.id.update);
       updateBtn.setOnClickListener(new View.OnClickListener() {
          public void onClick(View view) {
             updateNotification();
          }
       });
    }
    protected void displayNotification() {
       Log.i("Start", "notification");

       /* Invoking the default notification service */
       NotificationCompat.Builder  mBuilder =
       new NotificationCompat.Builder(this);

       mBuilder.setContentTitle("New Message");
       mBuilder.setContentText("You've received new message.");
       mBuilder.setTicker("New Message Alert!");
       mBuilder.setSmallIcon(R.drawable.woman);

       /* Increase notification number every time a new notificatior
       mBuilder.setNumber(++numMessages);

       /* Creates an explicit intent for an Activity in your app */
       Intent resultIntent = new Intent(this, NotificationView.class

       TaskStackBuilder stackBuilder = TaskStackBuilder.create(this)
       stackBuilder.addParentStack(NotificationView.class);

       /* Adds the Intent that starts the Activity to the top of the
       stackBuilder.addNextIntent(resultIntent);
       PendingIntent resultPendingIntent =
          stackBuilder.getPendingIntent(
             0,
             PendingIntent.FLAG_UPDATE_CURRENT
          );
```

```
      mBuilder.setContentIntent(resultPendingIntent);

      mNotificationManager =
      (NotificationManager) getSystemService(Context.NOTIFICATION_S

      /* notificationID allows you to update the notification later
      mNotificationManager.notify(notificationID, mBuilder.build()
   }

   protected void cancelNotification() {
      Log.i("Cancel", "notification");
      mNotificationManager.cancel(notificationID);
   }

   protected void updateNotification() {
      Log.i("Update", "notification");

      /* Invoking the default notification service */
      NotificationCompat.Builder  mBuilder =
      new NotificationCompat.Builder(this);

      mBuilder.setContentTitle("Updated Message");
      mBuilder.setContentText("You've got updated message.");
      mBuilder.setTicker("Updated Message Alert!");
      mBuilder.setSmallIcon(R.drawable.woman);

     /* Increase notification number every time a new notification
      mBuilder.setNumber(++numMessages);

      /* Creates an explicit intent for an Activity in your app */
      Intent resultIntent = new Intent(this, NotificationView.class

      TaskStackBuilder stackBuilder = TaskStackBuilder.create(this)
      stackBuilder.addParentStack(NotificationView.class);

      /* Adds the Intent that starts the Activity to the top of the
      stackBuilder.addNextIntent(resultIntent);
      PendingIntent resultPendingIntent =
         stackBuilder.getPendingIntent(
            0,
            PendingIntent.FLAG_UPDATE_CURRENT
         );

      mBuilder.setContentIntent(resultPendingIntent);

      mNotificationManager =
      (NotificationManager) getSystemService(Context.NOTIFICATION_S

      /* Update the existing notification using same notification I
      mNotificationManager.notify(notificationID, mBuilder.build()
   }
}
```

以下是修改的主活动文件的内容 src/com.yiibai.notificationdemo/NotificationView.java.

```
package com.example.notificationdemo;

import android.os.Bundle;
import android.app.Activity;

public class NotificationView extends Activity{
   @Override
   public void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.notification);
   }

}
```

下面文件 res/layout/activity_main.xml 的内容如下:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <Button android:id="@+id/start"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/start_note"/>

   <Button android:id="@+id/cancel"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/cancel_note" />

   <Button android:id="@+id/update"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/update_note" />

</LinearLayout>
```

下面是 res/layout/notification.xml 文件的内容：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"    >
   <TextView
   android:layout_width="fill_parent"
   android:layout_height="400dp"
   android:text="Hi, Your Detailed notification view goes here....'
</LinearLayout>
```

下面文件 res/values/strings.xml 的内容中定义两个新的常量：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">NotificationDemo</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="start_note">Start Notification</string>
    <string name="cancel_note">Cancel Notification</string>
    <string name="update_note">Update Notification</string>

</resources>
```

下面是 AndroidManifest.xml 文件的内容：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/androic
    package="com.yiibai.notificationdemo"
    android:versionCode="1"
    android:versionName="1.0" >

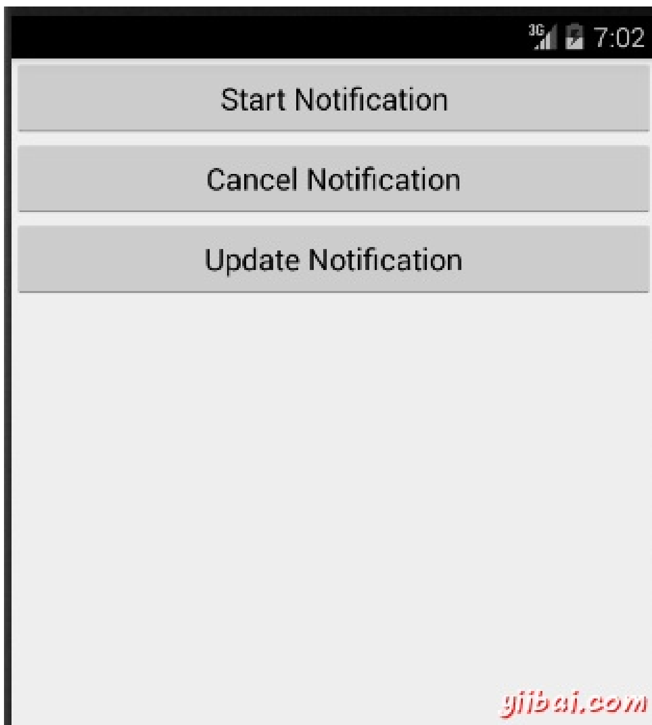
    <uses-sdk
        android:minSdkVersion="17"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.yiibai.notificationdemo.MainActivity'
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" /

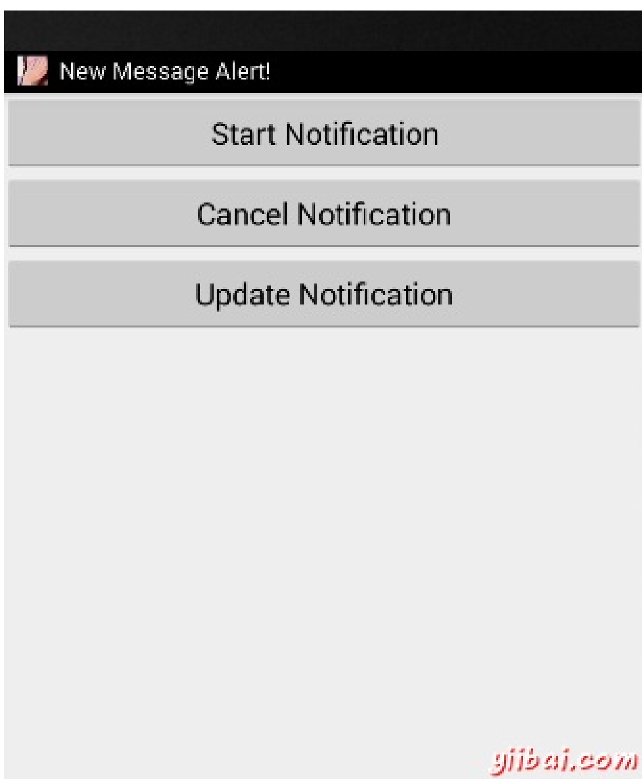
                <category android:name="android.intent.category.LAU
            </intent-filter>
        </activity>
        <activity android:name=".NotificationView"
             android:label="Details of notification"
             android:parentActivityName=".MainActivity">
       <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".MainActivity"/>
        </activity>
    </application>

</manifest>
```

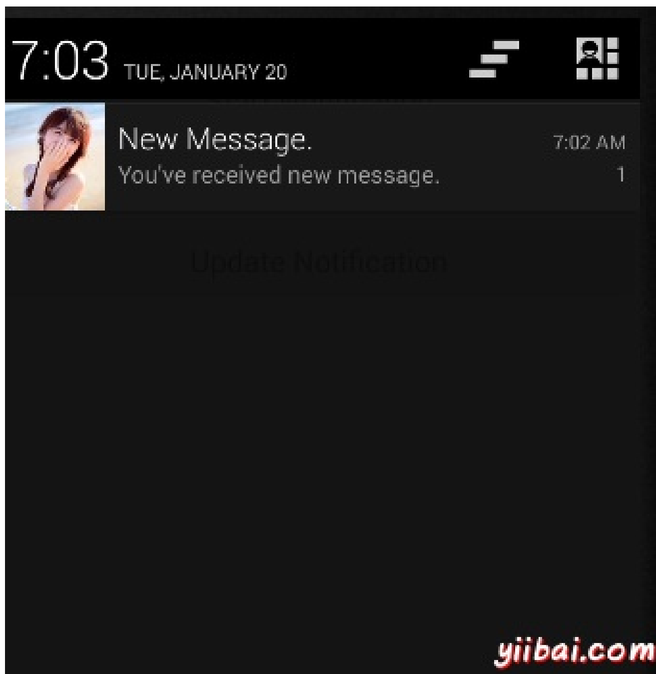
我们尝试运行NotificationDemo 应用程序。AVD安装的应用程序，并启动它，如果一切设置和应用都没有问题，它会显示以下模拟器窗口：



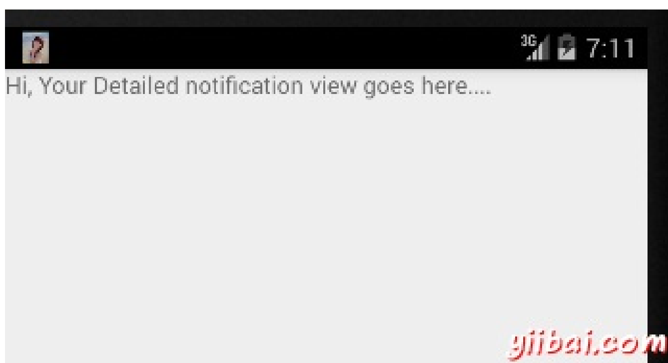
现在单击“Start Notification”通知按钮，会看到在上面的一条消息“New Message Alert!”将短暂显示后，将有下面的屏幕左上角有一个小图标。



现在，让我们展开视图，长按小图标，一秒钟后它会显示日期信息，这是时间的时候，应该释放鼠标拖动状态栏的情况下。会看到状态栏将扩大，会得到以下画面：



现在，让我们尝试在图像上点击图标，这将启动新的活动，已设置使用的意图，将有以下屏幕：



接下来，可以点击“Detail of notification”，将带回到主屏幕，可以尝试使用更新通知按钮，将更新现有的通知和数量将增加1，但如果发送通知，新的通知ID会继续增加在堆栈中，会看到他们在屏幕上单独列示。

图查看大图通知

下面的代码片断演示了如何改变的通知，上面代码中创建使用收件箱大视图样式。要更新 displayNotification() 方法来显示这个功能：

```
protected void displayNotification() {
   Log.i("Start", "notification");

   /* Invoking the default notification service */
   NotificationCompat.Builder  mBuilder =
   new NotificationCompat.Builder(this);

   mBuilder.setContentTitle("New Message");
   mBuilder.setContentText("You've received new message.");
```

```
        mBuilder.setTicker("New Message Alert!");
        mBuilder.setSmallIcon(R.drawable.woman);

        /* Increase notification number every time a new notificatior
        mBuilder.setNumber(++numMessages);

        /* Add Big View Specific Configuration */
        NotificationCompat.InboxStyle inboxStyle =
               new NotificationCompat.InboxStyle();

        String[] events = new String[6];
        events[0] = new String("This is first line....");
        events[1] = new String("This is second line...");
        events[2] = new String("This is third line...");
        events[3] = new String("This is 4th line...");
        events[4] = new String("This is 5th line...");
        events[5] = new String("This is 6th line...");

        // Sets a title for the Inbox style big view
        inboxStyle.setBigContentTitle("Big Title Details:");
        // Moves events into the big view
        for (int i=0; i < events.length; i++) {

           inboxStyle.addLine(events[i]);
        }
        mBuilder.setStyle(inboxStyle);

        /* Creates an explicit intent for an Activity in your app */
        Intent resultIntent = new Intent(this, NotificationView.class

        TaskStackBuilder stackBuilder = TaskStackBuilder.create(this)
        stackBuilder.addParentStack(NotificationView.class);

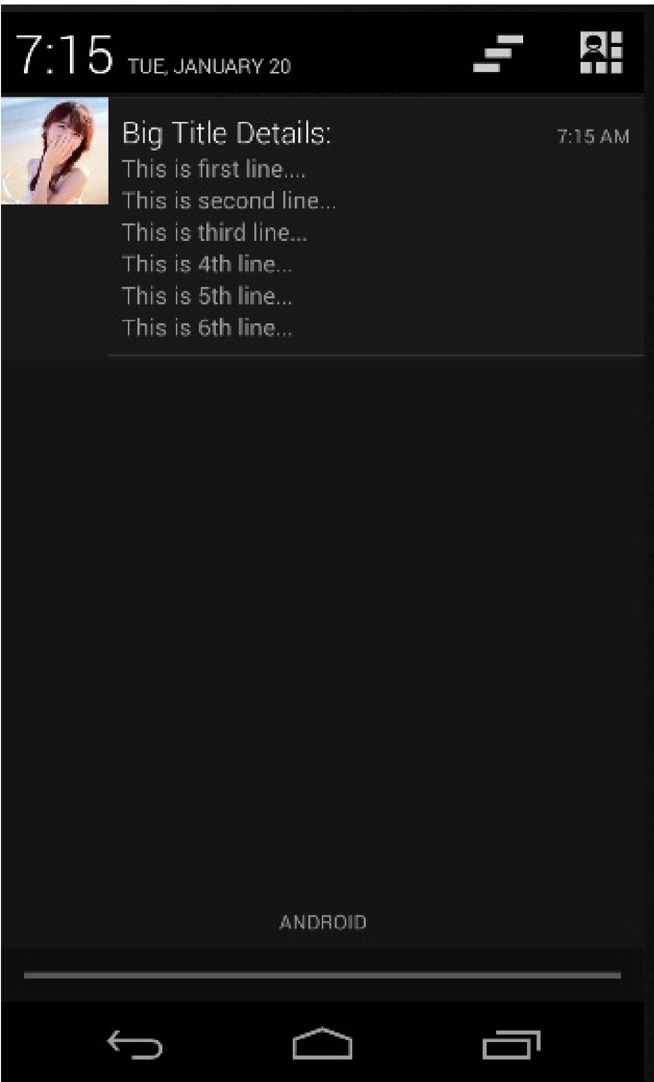
        /* Adds the Intent that starts the Activity to the top of the
        stackBuilder.addNextIntent(resultIntent);
        PendingIntent resultPendingIntent =
           stackBuilder.getPendingIntent(
              0,
              PendingIntent.FLAG_UPDATE_CURRENT
           );

        mBuilder.setContentIntent(resultPendingIntent);

        mNotificationManager =
        (NotificationManager) getSystemService(Context.NOTIFICATION_S

        /* notificationID allows you to update the notification later
        mNotificationManager.notify(notificationID, mBuilder.build()
     }
```

现在，如果尝试运行应用程序，然后会发现下面的结果视图的扩展形式：


7:15 TUE, JANUARY 20
Big Title Details:
7:15 AM
This is first line....
This is second line...
This is third line...
This is 4th line...
This is 5th line...
This is 6th line...
ANDROID

Android基于位置服务 - Android开发教程

Android 的位置API，很容易让创建位置感知的应用程序，而不需要把重点放在相关定位技术细节。这在谷歌服务的帮助下有利于应用程序添加位置感知，自动定位跟踪，地理和活动识别成为可能。

本教程介绍了如何使用位置服务在应用程序来获取当前的位置，得到周期性位置更新，查找地址等

Location 对象

Location对象代表一个地理位置可包括纬度，经度，时间戳和其它信息，如重力，高度和速度。有以下重要的方法在使用Location对象位置的具体信息：

S.N.	方法和说明
1	**float distanceTo(Location dest)** 返回在这个位置，并在给定的位置之间大致距离（单位：米）
2	**float getAccuracy()** 得到这个位置的估计精度，以米为单位
3	**double getAltitude()** （如果可用）获取的高度，如：海拔（单位：米）
4	**float getBearing()** 获取轴承，以度为单位
5	**double getLatitude()** 获得纬度，单位为度
6	**double getLongitude()**得到经度，单位为度
7	**float getSpeed()** 获取速度（如果可用），在地上以米/秒
8	**boolean hasAccuracy()** 如果此位置有一个精确度
9	**boolean hasAltitude()** True - 如果此位置有一个高度
10	**boolean hasBearing()** True 如果该位置有一个支撑
11	**boolean hasSpeed()** True如果这个位置有一个速度
12	**void reset()** 清除单元内容
13	**void setAccuracy(float accuracy)** 设置此位置的估计精度（米）
14	**void setAltitude(double altitude)** 设置海拔高度（米）
15	**void setBearing(float bearing)** 设置支承，以度为单位
16	**void setLatitude(double latitude)** 设置的纬度，单位为度
17	**void setLongitude(double longitude)** 设置的经度，单位为度
18	**void setSpeed(float speed)** 设置速度，在地上以米/秒
19	**String toString()** 返回包含此对象的简洁，可读的描述字符串信息

当前位置

要获得目前的位置，需要创建一个位置的 LocationClient 对象，将它连接到位置服务使用connect()方法，然后调用其 getLastLocation() 方法。此方法返回最近的位置Location对象的形式：其中包含纬度和经度坐标和其他信息，如上面所述。要在活动中有基于位置的功能，将需要实现两个接口：

- GooglePlayServicesClient.ConnectionCallbacks
- GooglePlayServicesClient.OnConnectionFailedListener

这些接口提供了以下重要的，需要在活动类实现回调方法：

S.N.	回调方法及说明
1	**abstract void onConnected(Bundle connectionHint)** 这个回调方法被调用时，位置服务是成功连接到客户端的位置。将使用connect()方法来连接到客户端的位置
2	**abstract void onDisconnected()** 当客户端断开这个回调方法被调用。将使用disconnect()方法从位置客户端断开连接
3	**abstract void onConnectionFailed(ConnectionResult result)** 这个回调方法被调用时，有客户端连接到服务的错误

> 应该创在建客户活动类onCreate()方法中调用猎取位置，然后将其连接在onStart()，让位置服务维持目前的位置，而活动是完全可见。断开客户端onStop()方法，这样应用程序是不可见的，位置服务不是维持在目前的位置。这在很大程度上有助于节省电池电量。

获取更新位置

如果愿意的位置更新，那么除了上面提到的接口，需要实现LocationListener 接口，该接口提供了以下回调方法，需要在活动类实现：

S.N.	回调方法说明
1	**abstract void onLocationChanged(Location location)** 此回调方法用于从LocationClient接收通知时的位置发生了变化

位置服务质量

LocationRequest对象的位置从LocationClient更新服务质量（QoS），setter方法可以用它来处理QoS。

S.N.	方法 & 描述
1	**setExpirationDuration(long millis)** 设置此请求的时间，以毫秒为单位
2	**setExpirationTime(long millis)** 自启动设置请求过期时间，单位为毫秒
3	**setFastestInterval(long millis)** 明确设置最快的时间间隔位置更新，以毫秒为单位
4	**setInterval(long millis)** 设置为主动位置更新，以毫秒为单位所需的时间间隔
5	**setNumUpdates(int numUpdates)** 设置的位置更新次数
6	**setPriority(int priority)** 设置请求的优先级

例如，如果应用需要较准确位置，它应该创建一个位置请求 setPriority(int) 设置 PRIORITY_HIGH_ACCURACY和setInterval(long) 为5秒。还可以使用更大的间隔和/或其他优先如PRIORITY_LOW_POWER 要求“城市” 的级别精度或 PRIORITY_BALANCED_POWER_ACCURACY为“块”级别的精度。

> 活动应该考虑删除在所有位置请求进入后台时（例如 onPause()），或者至少交换请求到一个更大的间隔和降低质量以节省电力消耗。

显示位置地址

Location对象可以使用 Geocoder.getFromLocation() 方法来获得一个地址，对于一个给定的纬度和经度。这种方法是同步的，可能要花很长的时间来完成其工作，所以应该调用AsyncTask类的 doInBackground() 方法。

AsyncTask必须要使用的子类和子类会覆盖 doInBackground（Params....）方法中执行任务UI线程上的后台计算完成后，当 onPostExecute(Result) 方法被调用显示结果。还有一个更重要的方法在AyncTask execute(Params... params)，此方法使用指定的参数执行任务。

检查下面的例子中，我们如何使用 AynchTask 在任何 Android 应用程序完成工作的主要任务，而不会干扰后台。

示例

下面的示例演示如何在实际使用位置服务在应用程序来获取当前位置、等效地址等等。这个例子中，需要实际配备最新的 Android OS 移动设备，否则仿真器可能无法正常工作。

安装谷歌播放服务**SDK**

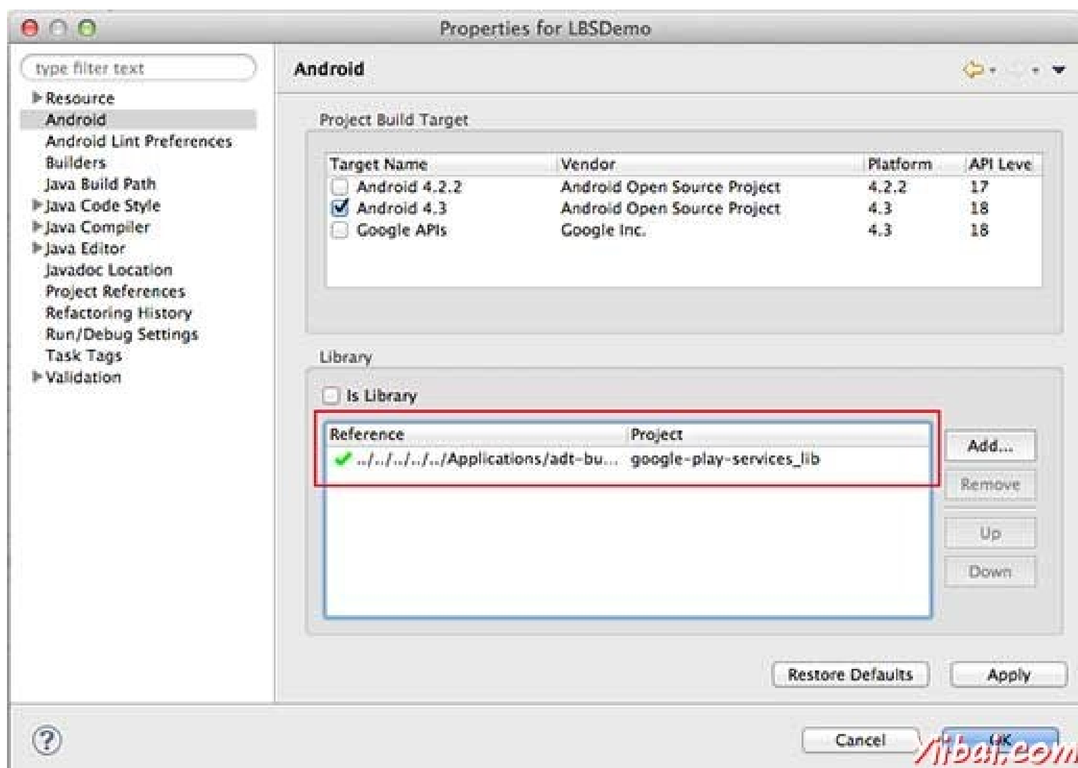
在继续之前有位置在Android应用程序的支持，NEET设置谷歌播放服务SDK使用以下简单的步骤：

步骤	描述
1	启动SDK管理器 在Eclipse（使用ADT），选择 Window > Android SDK Manager 在Windows中，双击SDK Manager.exe 请通过在Android SDK目录的根目录 在Mac或Linux打开一个终端并导航到工具/目录下在Android SDK目录下，然后执行Android SDK
2	搜索**Google Play services**从给定的软件包列表服务选项下的**Extra**，如果没有安装它，那么进行安装。在谷歌Play业务SDK是在保存在Android SDK环境*<android-sdk>/extras/google/google_play_services/*
3	复制库项目在*<android-sdk>/extras/google/google_play_services/libproject/google-play-services_lib/*到维护Android应用项目的位置。如果使用的是Eclipse，导入库项目到工作区。点击 **File > Import**, 选择 **Android > Existing Android Code** 到工作区, 并浏览到 *<android-sdk>/extras/google/google_play_services/libproject/*, 库项目将其导入

创建**Android**应用程序

步骤	描述
1	使用Eclipse IDE创建Android应用程序，并将其命名为：LBSDemo，在创建这个项目时请确保目标SDK 编译在Android SDK的最新版本或使用更高级别的API
2	添加 **Google Play**服务库中的项目按照以下给出简单的步骤
3	修改 *src/MainActivity.java* 文件，并添加所需的代码如下所示采取获取当前位置和它的等效转交地址
4	修改布局XML文件*res/layout/activity_main.xml* 添加所有GUI组件，其中包括三个按钮和两个文本视图来显示位置/地址
5	修改*res/values/strings.xml*定义所需的常量值
6	修改 *AndroidManifest.xml* 如下所示
7	运行该应用程序启动Android模拟器和验证应用程序所做的修改结果

让我们在项目中添加引用谷歌Play服务。右键单击该项目，并选择Build Path > Configure Build Path > Android >，然后单击“Add ”按钮，将显示要添加的谷歌Play service_liboption，只要双击就可以了，这将增加所需的库的引用，将有窗口如下所示：



以下是内容的修改主活动文件 src/com.yiibai.lbsdemo/MainActivity.java

```
package com.example.lbsdemo;

import java.io.IOException;
import java.util.List;
import java.util.Locale;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GooglePlayServicesClient;
import com.google.android.gms.location.LocationClient;

import android.content.Context;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;

import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends FragmentActivity implements
GooglePlayServicesClient.ConnectionCallbacks,
GooglePlayServicesClient.OnConnectionFailedListener
{
   LocationClient mLocationClient;
```

```
    private TextView addressLabel;
    private TextView locationLabel;
    private Button getLocationBtn;
    private Button disconnectBtn;
    private Button connectBtn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);

       locationLabel = (TextView) findViewById(R.id.locationLabel);
       addressLabel = (TextView) findViewById(R.id.addressLabel);
       getLocationBtn = (Button) findViewById(R.id.getLocation);

       getLocationBtn.setOnClickListener(new View.OnClickListener()
          public void onClick(View view) {
             displayCurrentLocation();
          }
       });
       disconnectBtn = (Button) findViewById(R.id.disconnect);
       disconnectBtn.setOnClickListener(new View.OnClickListener() {
          public void onClick(View view) {
             mLocationClient.disconnect();
             locationLabel.setText("Got disconnected....");
          }
       });
       connectBtn = (Button) findViewById(R.id.connect);
       connectBtn.setOnClickListener(new View.OnClickListener() {
          public void onClick(View view) {
             mLocationClient.connect();
             locationLabel.setText("Got connected....");
          }
       });
       // Create the LocationRequest object
       mLocationClient = new LocationClient(this, this, this);
    }
    @Override
    protected void onStart() {
       super.onStart();
       // Connect the client.
       mLocationClient.connect();
       locationLabel.setText("Got connected....");
    }
    @Override
    protected void onStop() {
       // Disconnect the client.
       mLocationClient.disconnect();
       super.onStop();
       locationLabel.setText("Got disconnected....");
    }
    @Override
    public void onConnected(Bundle dataBundle) {
```

```
      // Display the connection status
      Toast.makeText(this, "Connected", Toast.LENGTH_SHORT).show();
   }
   @Override
   public void onDisconnected() {
      // Display the connection status
      Toast.makeText(this, "Disconnected. Please re-connect.",
      Toast.LENGTH_SHORT).show();
   }
   @Override
   public void onConnectionFailed(ConnectionResult connectionResult
      // Display the error code on failure
      Toast.makeText(this, "Connection Failure : " +
      connectionResult.getErrorCode(),
      Toast.LENGTH_SHORT).show();
   }
   public void displayCurrentLocation() {
      // Get the current location's latitude & longitude
      Location currentLocation = mLocationClient.getLastLocation();
      String msg = "Current Location: " +
      Double.toString(currentLocation.getLatitude()) + "," +
      Double.toString(currentLocation.getLongitude());

      // Display the current location in the UI
      locationLabel.setText(msg);

      // To display the current address in the UI
      (new GetAddressTask(this)).execute(currentLocation);
   }
   /*
    * Following is a subclass of AsyncTask which has been used to g
    * address corresponding to the given latitude & longitude.
    */
   private class GetAddressTask extends AsyncTask<Location, Void, S
      Context mContext;
      public GetAddressTask(Context context) {
         super();
         mContext = context;
      }

      /*
       * When the task finishes, onPostExecute() displays the addre
       */
      @Override
      protected void onPostExecute(String address) {
         // Display the current address in the UI
         addressLabel.setText(address);
      }
      @Override
      protected String doInBackground(Location... params) {
         Geocoder geocoder =
         new Geocoder(mContext, Locale.getDefault());
         // Get the current location from the input parameter list
```

```
            Location loc = params[0];
            // Create a list to contain the result address
            List<Address> addresses = null;
            try {
               addresses = geocoder.getFromLocation(loc.getLatitude(),
               loc.getLongitude(), 1);
            } catch (IOException e1) {
               Log.e("LocationSampleActivity",
               "IO Exception in getFromLocation()");
               e1.printStackTrace();
               return ("IO Exception trying to get address");
            } catch (IllegalArgumentException e2) {
               // Error message to post in the log
               String errorString = "Illegal arguments " +
               Double.toString(loc.getLatitude()) +
               " , " +
               Double.toString(loc.getLongitude()) +
               " passed to address service";
               Log.e("LocationSampleActivity", errorString);
               e2.printStackTrace();
               return errorString;
            }
            // If the reverse geocode returned an address
            if (addresses != null && addresses.size() > 0) {
               // Get the first address
               Address address = addresses.get(0);
               /*
               * Format the first line of address (if available),
               * city, and country name.
               */
               String addressText = String.format(
               "%s, %s, %s",
               // If there's a street address, add it
               address.getMaxAddressLineIndex() > 0 ?
               address.getAddressLine(0) : "",
               // Locality is usually a city
               address.getLocality(),
               // The country of the address
               address.getCountryName());
               // Return the text
               return addressText;
            } else {
               return "No address found";
            }
         }
      }// AsyncTask class
}
```

Following will be the content of res/layout/activity_main.xml file:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <Button android:id="@+id/getLocation"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/get_location"/>

   <Button android:id="@+id/disconnect"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/disconnect"/>

   <Button android:id="@+id/connect"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/connect"/>

    <TextView
   android:id="@+id/locationLabel"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"/>

   <TextView
   android:id="@+id/addressLabel"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"/>

</LinearLayout>
```

下面文件 res/values/strings.xml 内容中定义两个新的常量：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">LBSDemo</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="get_location">Get Location</string>
    <string name="disconnect">Disconnect Service</string>
    <string name="connect">Connect Service</string>
</resources>
```

以下是 AndroidManifest.xml 文件中默认的内容：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/androic
    package="com.yiibai.lbsdemo"
    android:versionCode="1"
    android:versionName="1.0" >

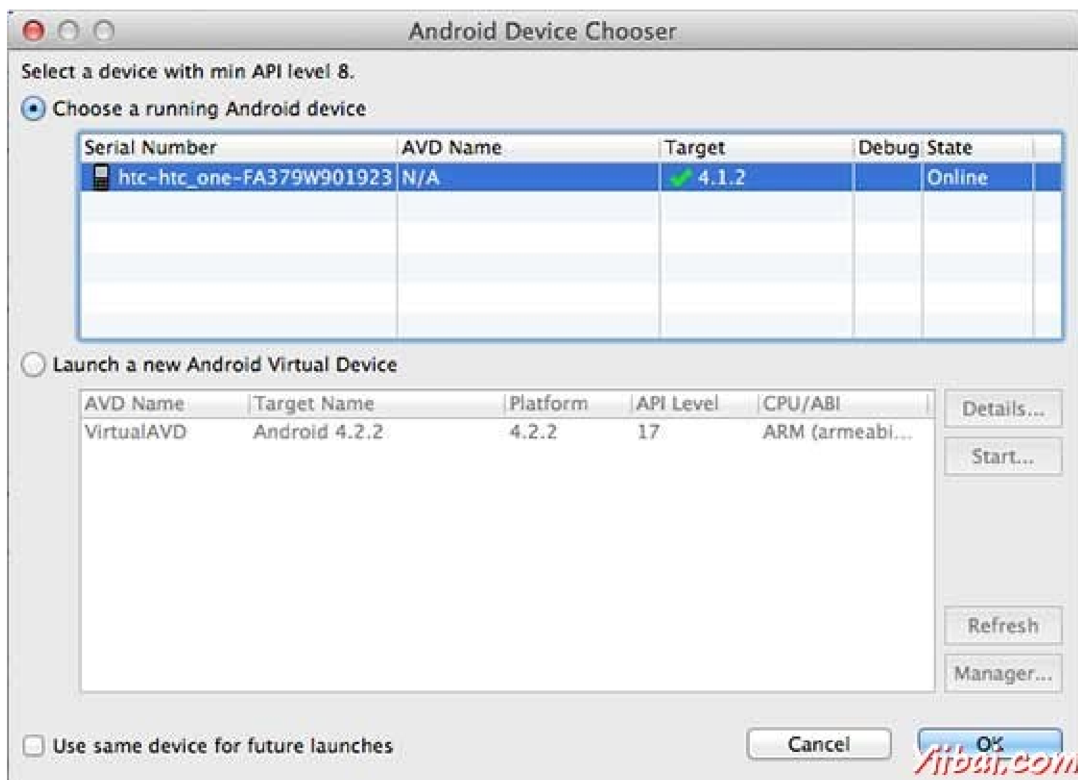
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />
   <uses-permission android:name="android.permission.ACCESS_COARSE_

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.yiibai.lbsdemo.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" /

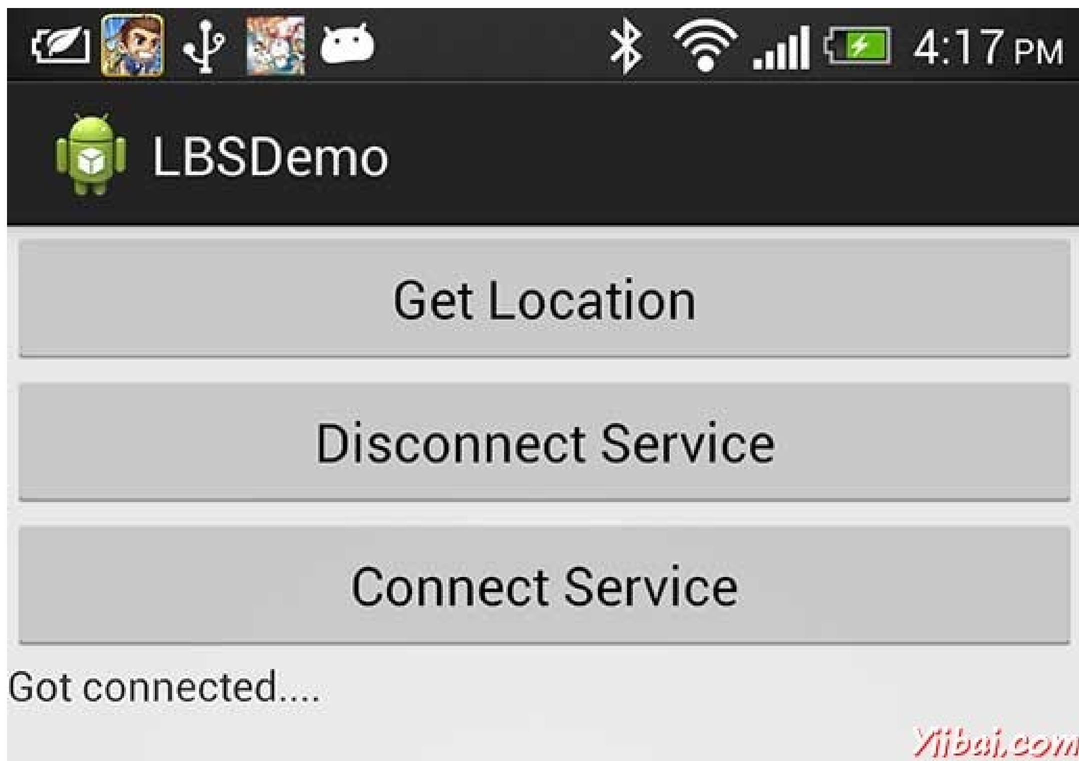
                <category android:name="android.intent.category.LAU
            </intent-filter>
        </activity>
    </application>

</manifest>
```

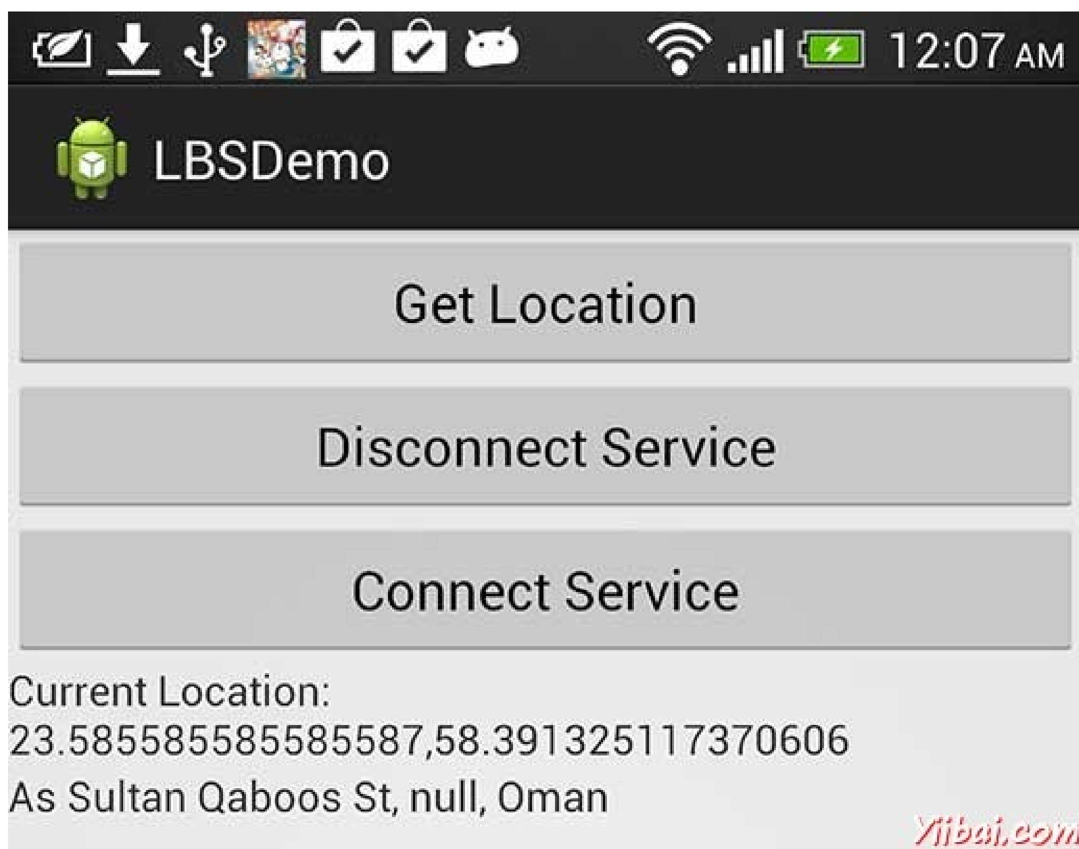
我们尝试运行LBSDemo应用程序。 Eclipse AVD安装的应用程序，并启动它，如果一切设置和应用都没有问题，它会显示以下模拟器窗口：



选择移动设备作为一个选项，然后检查移动设备，这将显示以下画面：



现在看到的位置选择位置"按钮，将显示位置信息如下：



可以尝试断开位置，客户端使用服务，然后断开连接使用连接服务按钮，还可以修改位置更新，如上所述，可查阅 Android官方文档。

Android发送电子邮件 - Android开发教程

在前面已经学会了 Android 的意图(Intent)，这是落实意图，即一个对象。来自一个部件的消息传递到另一个组件使用 - 在应用程序或应用程序之外。

因此这里不需要从头开始，因为它们已经可以像 Gmail 和 K9mail 开发电子邮件客户端。但需要从 Android 应用程序发送的电子邮件，编写一个活动Activity，使用Android设备发送电子邮件需要启动电子邮件客户端并发送电子邮件。为了这个目的，活动将伴随着相应的数据负载一个ACTION_SEND发送到 Android 意图解析器。指定选择器提供适当的接口供用户选择如何发送电子邮件数据。

以下部分说明 Intent 对象发送电子邮件。

Intent 对象 - 动作发送电子邮件

使用ACTION_SEND 的动作启动 Android 设备上安装一个电子邮件客户端。以下是简单的语法创建一个Intent 用ACTION_SEND动作

```
Intent emailIntent = new Intent(Intent.ACTION_SEND);
```

Intent 对象 - 数据/发送电子邮件的类型

要发送电子邮件，需要指定mailto：URI使用 setData() 方法并且数据类型是text/plain使用settype()方法如下：

```
emailIntent.setData(Uri.parse("mailto:"));
emailIntent.setType("text/plain");
```

Intent 对象- 附加发送电子邮件

Android已经内置支持TO, SUBJECT, CC, TEXT等域，可以在附加 Intent 之前发送到目标的电子邮件客户端的Intent。可以使用额外的字段后电子邮件：

S.N.	额外数据 & 描述
1	**EXTRA_BCC** String[] 持有应密件复制电子邮件地址
2	**EXTRA_CC** String[] 持有复制电子邮件地址
3	**EXTRA_EMAIL** String[] 持有应递送到电子邮件地址
4	**EXTRA_HTML_TEXT** 与该意图相关联的常数字符串，使用 ACTION_SEND 替代 EXTRA_TEXT 为 HTML 格式的文本
5	**EXTRA_SUBJECT** 常量字符串持有一条消息的所需主题行
6	**EXTRA_TEXT** 与该意图相关联的CharSequence常量，具有 ACTION_SEND用来提供文字数据被发送
7	**EXTRA_TITLE** 一个CharSequence对话框的标题，提供给用户在 ACTION_CHOOSER使用时

下面是一个例子展示如何分配额外的数据到 intent

```
emailIntent.putExtra(Intent.EXTRA_EMAIL  , new String[]{"recipient@
emailIntent.putExtra(Intent.EXTRA_SUBJECT, "subject of email");
emailIntent.putExtra(Intent.EXTRA_TEXT   , "body of email");
```

示例

下面的示例演示如何在实际使用Intent对象启动电子邮件客户端发送电子邮件给定的收件人。要测试这个例子，需要实际配备了最新的Android OS的移动设备，否则仿真器可能无法正常工作。其次，需要在您的设备上安装一个电子邮件客户端，如 Gmail 或 K9mail

步骤	描述
1	使用Android Studio创建Android应用程序，并将它命名为 SendEmailDemounde。创建这个项目，确保目标SDK并编译在Android SDK为最新版本以及使用更高级别的API
2	修改 src/MainActivity.java 文件，并添加所需的代码，以发送电子邮件
3	修改所需的布局XML文件*res/layout/activity_main.xml* 添加GUI组件。这里添加一个简单的按钮，启动电子邮件客户端
4	修改*res/values/strings.xml*定义所需的常量值
5	修改 AndroidManifest.xml 如下所示
6	运行该应用程序启动 Android模拟器并验证应用程序所做的修改结果。

以下是修改的主活动文件的内容 src/com.yiibai.sendemaildemo/MainActivity.java.

```
package com.example.sendemaildemo;

import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      Button startBtn = (Button) findViewById(R.id.sendEmail);
      startBtn.setOnClickListener(new View.OnClickListener() {
         public void onClick(View view) {
         sendEmail();
      }
   });

   }
   protected void sendEmail() {
      Log.i("Send email", "");

      String[] TO = {"amrood.admin@gmail.com"};
      String[] CC = {"mcmohd@gmail.com"};
      Intent emailIntent = new Intent(Intent.ACTION_SEND);
      emailIntent.setData(Uri.parse("mailto:"));
      emailIntent.setType("text/plain");

      emailIntent.putExtra(Intent.EXTRA_EMAIL, TO);
      emailIntent.putExtra(Intent.EXTRA_CC, CC);
      emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Your subject");
      emailIntent.putExtra(Intent.EXTRA_TEXT, "Email message goes h

      try {
         startActivity(Intent.createChooser(emailIntent, "Send mai
         finish();
         Log.i("Finished sending email...", "");
      } catch (android.content.ActivityNotFoundException ex) {
         Toast.makeText(MainActivity.this,
         "There is no email client installed.", Toast.LENGTH_SHORT
      }
   }
```

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
       // Inflate the menu; this adds items to the action bar if it
       getMenuInflater().inflate(R.menu.main, menu);
       return true;
    }
}
```

下面是文件 res/layout/activity_main.xml 的内容：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <Button android:id="@+id/sendEmail"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/compose_email"/>

</LinearLayout>
```

下面文件 res/values/strings.xml 的内容中定义两个新的常量：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">SendEmailDemo</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="compose_email">Compose Email</string>

</resources>
```

以下是文件 AndroidManifest.xml 默认的内容：

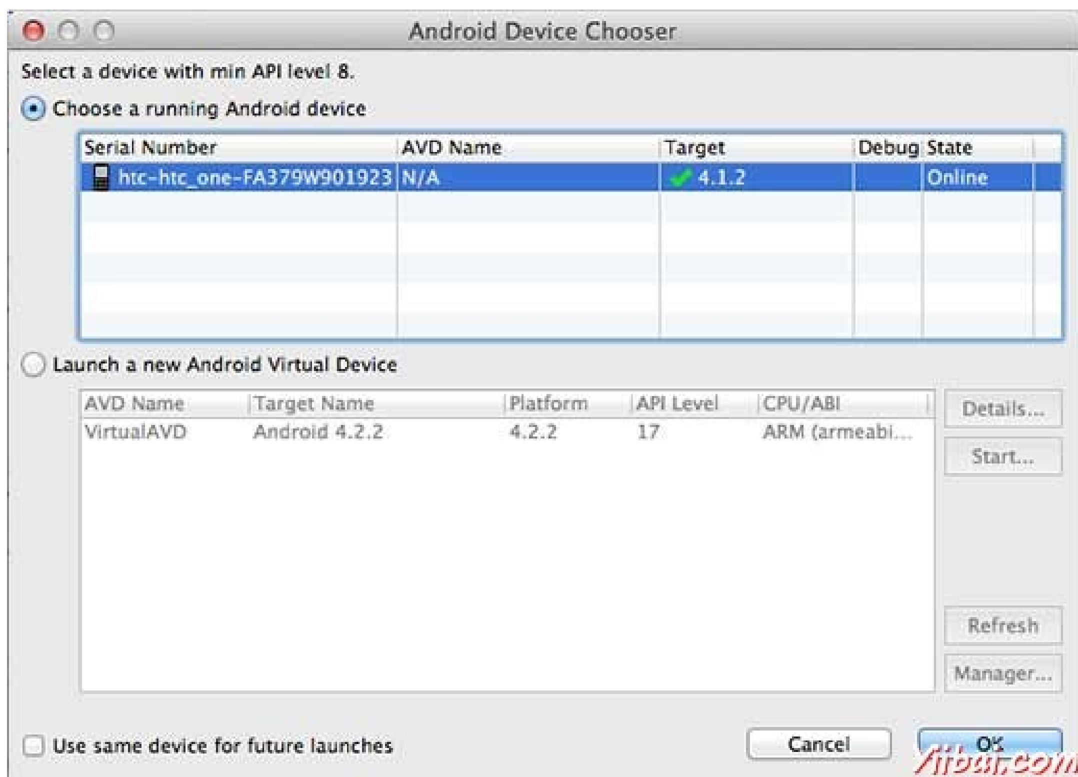
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/androic
    package="com.yiibai.sendemaildemo"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

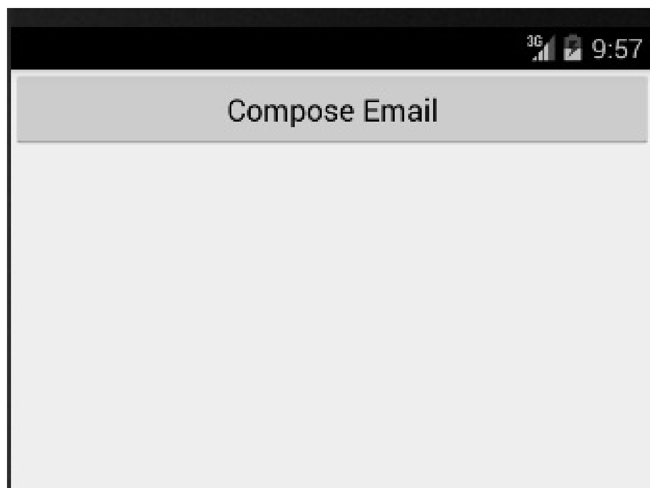
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.yiibai.sendemaildemo.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" /

                <category android:name="android.intent.category.LAU
            </intent-filter>
        </activity>
    </application>
</manifest>
```

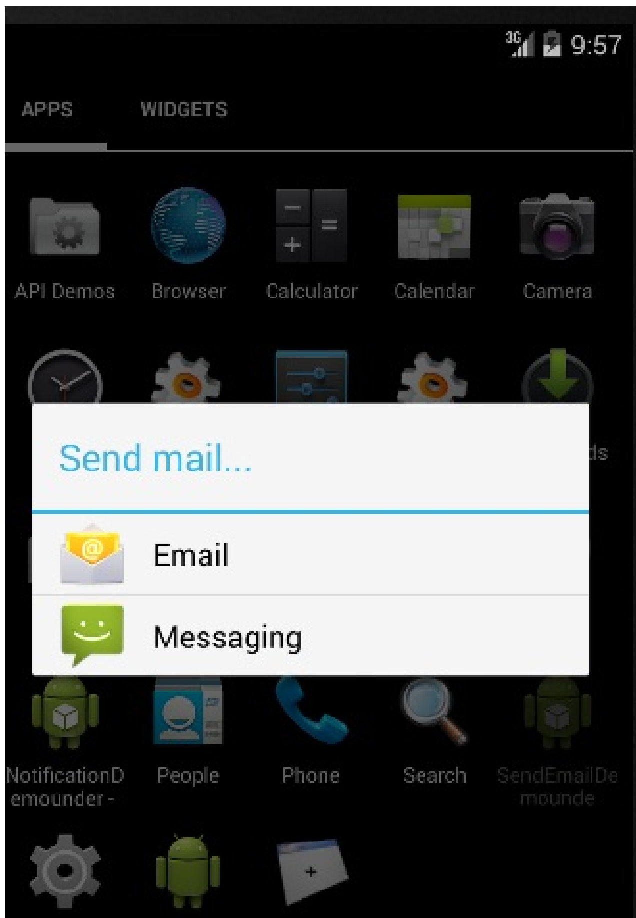
我们尝试运行SendEmailDemo 应用程序。Eclipse AVD安装的应用程序，并启动它，如果一切设置和应用都没有问题，它会显示以下模拟器窗口：



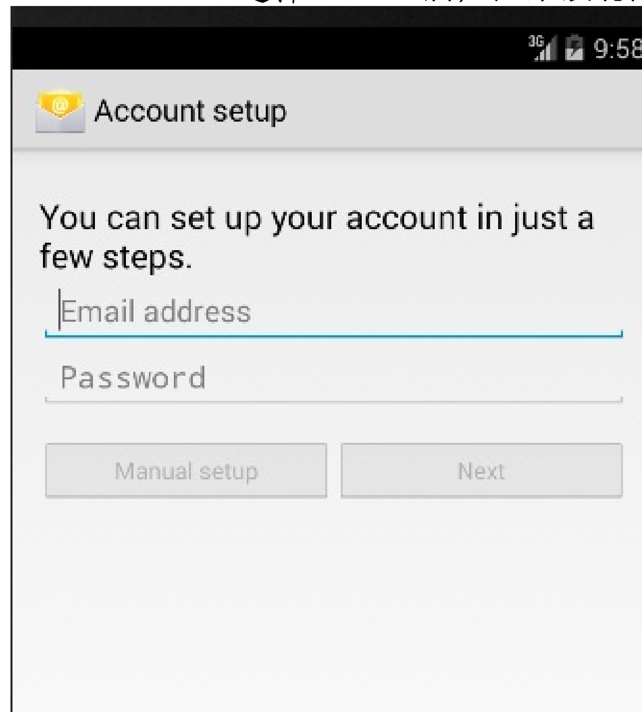
选择移动设备作为一个选项，然后检查移动设备，这将显示以下画面：



现在使用Compose Email"按钮，列出了所有已安装的电子邮件客户端。从列表中，可以选择其中的电子邮件客户端发送电子邮件。要使用Gmail客户端发送电子邮件，将所有提供的默认值的字段，如下图所示。在这里，From：将默认的电子邮件ID，已经为Android设备注册。



选择 “email"后，如果没有配置帐号信息，则提示配置帐号信息：



可以修改默认字段，最后使用“send email ”按钮（标有红色矩形）提到的收件人发送电子邮件。 代码下载地下：http://pan.baidu.com/s/1qW2X0hm

Android发送短信/SMS - Android开发教程

有以下两种方式来使用 Android 设备发送短信：

- 使用 SmsManager 发送短信
- 使用内置 Intent 发送短信

使用SmsManager 发送短信

SmsManager管理，例如在给定的移动设备将数据发送到的SMS操作。可以创建此对象调用用静态方法SmsManager.getDefault() 如下：

```
SmsManager smsManager = SmsManager.getDefault();
```

创建 SmsManager 对象之后，可以使用 sendDataMessage() 方法指定的手机号码发送短信，如下：

```
smsManager.sendTextMessage("phoneNo", null, "SMS text", null, null)
```

除了上述方法外，SmsManager类可供选择的其他几个重要的函数。下面列出了这些方法：

S.N.	方法和说明
1	**ArrayList<String> divideMessage(String text)** 这个方法把一个消息文本分成几个片段，最大不能大于短信大小
2	**static SmsManager getDefault()** 这个方法被用来获取 SmsManager 的默认实例
3	**void sendDataMessage(String destinationAddress, String scAddress, short destinationPort, byte[] data, PendingIntent sentIntent, PendingIntent deliveryIntent)** 这个方法被用来发送一个基于数据 SMS 到特定的应用程序的端口
4	**void sendMultipartTextMessage(String destinationAddress, String scAddress, ArrayList<String> parts, ArrayList<PendingIntent> sentIntents, ArrayList<PendingIntent> deliveryIntents)** 发送一个基于多部分文本短信
5	**void sendTextMessage(String destinationAddress, String scAddress, String text, PendingIntent sentIntent, PendingIntent deliveryIntent)** 发送基于文本的短信

示例

下面的示例演示如何在实际中使用 SmsManager 对象给定的手机号码发送短信。

> 要尝试这个例子中，需要实际配备了最新 Android OS 的移动设备，否则仿真器可能无法正常工作。

步骤	描述
1	使用Android Studio 创建Android应用程序，并将它命名为SendSMSDemounder。在创建这个项目，确保目标 SDK 编译在Android SDK 的最新版本或使用更高级别的API
2	修改 *src/MainActivity.java* 文件，并添加所需的代码以发送短信
3	修改所需的布局XML文件 *res/layout/activity_main.xml* 添加任何GUI组件。加入了一个简单的GUI以输入手机号码并短信发送，以及一个简单的按钮发送短信。
4	修改 *res/values/strings.xml* 定义所需的常量值
5	修改 *AndroidManifest.xml* 如下所示
6	运行该应用程序启动Android模拟器并验证应用程序所做的修改结果。

以下是修改的主活动文件 src/com.yiibai.sendsmsdemo/MainActivity.java 的内容

```
package com.example.sendsmsdemo;

import android.os.Bundle;
import android.app.Activity;
import android.telephony.SmsManager;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity {

   Button sendBtn;
   EditText txtphoneNo;
   EditText txtMessage;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      sendBtn = (Button) findViewById(R.id.btnSendSMS);
```

```
         txtphoneNo = (EditText) findViewById(R.id.editTextPhoneNo);
         txtMessage = (EditText) findViewById(R.id.editTextSMS);

         sendBtn.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
               sendSMSMessage();
            }
         });

      }
      protected void sendSMSMessage() {
         Log.i("Send SMS", "");

         String phoneNo = txtphoneNo.getText().toString();
         String message = txtMessage.getText().toString();

         try {
            SmsManager smsManager = SmsManager.getDefault();
            smsManager.sendTextMessage(phoneNo, null, message, null, r
            Toast.makeText(getApplicationContext(), "SMS sent.",
            Toast.LENGTH_LONG).show();
         } catch (Exception e) {
            Toast.makeText(getApplicationContext(),
            "SMS faild, please try again.",
            Toast.LENGTH_LONG).show();
            e.printStackTrace();
         }
      }
      @Override
      public boolean onCreateOptionsMenu(Menu menu) {
         // Inflate the menu; this adds items to the action bar if it
         getMenuInflater().inflate(R.menu.main, menu);
         return true;
      }
   }
```

下面是 **res/layout/activity_main.xml** 文件的内容：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical" >

   <TextView
   android:id="@+id/textViewPhoneNo"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/phone_label" />

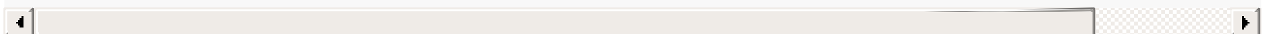
   <EditText
   android:id="@+id/editTextPhoneNo"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:inputType="phone"/>

   <TextView
   android:id="@+id/textViewMessage"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/sms_label" />

   <EditText
   android:id="@+id/editTextSMS"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:inputType="textMultiLine"/>

   <Button android:id="@+id/btnSendSMS"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/send_sms_label"/>

</LinearLayout>
```



下面文件 res/values/strings.xml 的内容中定义两个新的常量：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">SendSMSDemo</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="phone_label">Enter Phone Number:</string>
    <string name="sms_label">Enter SMS Message:</string>
    <string name="send_sms_label">Send SMS</string>

</resources>
```

以下是**AndroidManifest.xml** 文件的默认内容：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/androic
    package="com.yiibai.sendsmsdemo"
    android:versionCode="1"
    android:versionName="1.0" >

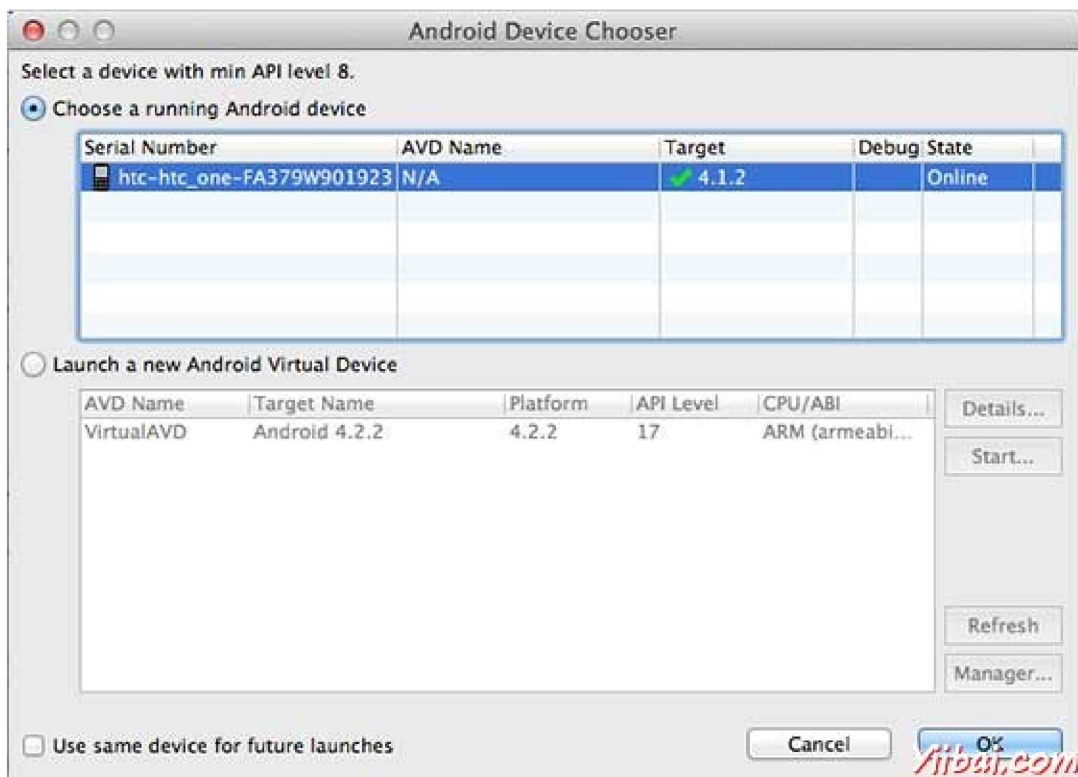
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />
    <uses-permission android:name="android.permission.SEND_SMS" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.yiibai.sendsmsdemo.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" /

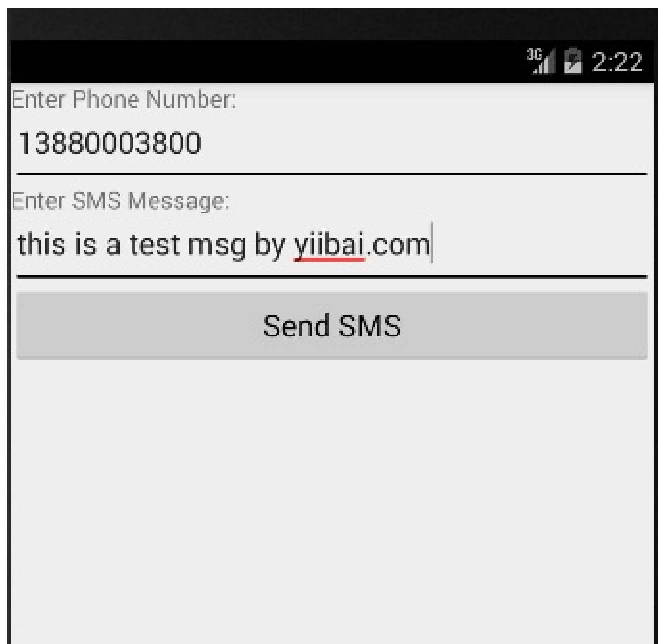
                <category android:name="android.intent.category.LAU
            </intent-filter>
        </activity>
    </application>

</manifest>
```

我们尝试运行 SendSMSDemo 应用程序。Eclipse的AVD上安装的应用程序，并启动它，如果一切的设置和应用代码都没有问题，它会显示以下模拟器窗口：



选择移动设备作为一个选项，然后检查移动设备，这将显示以下画面：



现在可以输入手机号码及文本消息并发送。最后点击"Send SMS"按钮发送短信。请确保GSM连接工作正常，以及提供正确的短信收件人。

可以把一些短信用逗号分隔，在程序中把它解析为一个数组的字符串，最后可以使用一个循环来发送消息给所有给定的手机号码。下一节将学习如何使用现有的 SMS 客户端发送短信。

使用内置Intent发送短信

发送短信通过调用Android内置短信功能，可以使用Android的Intent。以下部分说明使用 Intent 对象发送短信的功能。

Intent对象 - 发送短信动作

使用ACTION_VIEW 动作启动 Android 设备上安装 SMS 客户端。以下是简单的语法来创建一个 Intent 来使用 ACTION_VIEW 动作

```
Intent smsIntent = new Intent(Intent.ACTION_VIEW);
```

Intent对象 - 数据/发送短信类型

要发送的短信需要使用SetData()方法指定 smsto: 作为URI和数据类型将使用 setType() 方法如下vnd.android-dir/mms-sms：

```
smsIntent.setData(Uri.parse("smsto:"));
smsIntent.setType("vnd.android-dir/mms-sms");
```

Intent 对象- 附加发送短信

Android已经内置支持添加电话号码和短信发送短信如下：

```
smsIntent.putExtra("address"  , new String("0123456789;3393993300")
smsIntent.putExtra("sms_body"  , "Test SMS to Angilla");
```

> 这里address 和sms_body是大小写敏感的，应以小字符指定。可以指定一个以上的号码在单串，但由分号（;) 隔开。

示例

下面的示例演示如何在实际使用Intent对象启动SMS客户端发送短信给定的收件人。

> 要尝试这个例子中，需要实际配备了最新的 Android OS的移动设备，否则仿真器可能无法正常工作。

步骤	描述
1	使用Android Studio创建Android应用程序，并将它命名为SendSMSDemounder，创建这个项目，确保目标 SDK编译在Android SDK的最新版本或使用更高级别的API。
2	修改*src/MainActivity.java*文件，并添加所需的代码，以发送短信。
3	修改所需的布局XML文件 *res/layout/activity_main.xml* 添加任何GUI组件。添加一个简单的按钮用来触发启动SMS客户端。
4	修改 *res/values/strings.xml* 定义所需的常量值
5	修改 AndroidManifest.xml 如下所示
6	运行该应用程序启动Android模拟器并验证应用程序所做的修改结果。

以下是修改主活动文件 src/com.yiibai.sendsmsdemo/MainActivity.java 的内容

```
package com.example.sendsmsdemo;

import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      Button startBtn = (Button) findViewById(R.id.sendSMS);
      startBtn.setOnClickListener(new View.OnClickListener() {
         public void onClick(View view) {
         sendSMS();
      }
   });

   }
   protected void sendSMS() {
      Log.i("Send SMS", "");

      Intent smsIntent = new Intent(Intent.ACTION_VIEW);
      smsIntent.setData(Uri.parse("smsto:"));
```

```
      smsIntent.setType("vnd.android-dir/mms-sms");

      smsIntent.putExtra("address"  , new String ("0123456789"));
      smsIntent.putExtra("sms_body"  , "Test SMS to Angilla");
      try {
         startActivity(smsIntent);
         finish();
         Log.i("Finished sending SMS...", "");
      } catch (android.content.ActivityNotFoundException ex) {
         Toast.makeText(MainActivity.this,
         "SMS faild, please try again later.", Toast.LENGTH_SHORT).
      }
   }
   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if it
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
   }
}
```

下面是 **res/layout/activity_main.xml** 文件的内容：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <Button android:id="@+id/sendSMS"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/compose_sms"/>

</LinearLayout>
```

下面文件 res/values/strings.xml 的内容中定义两个新的常量：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">SendSMSDemo</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="compose_sms">Compose SMS</string>

</resources>
```

以下是**AndroidManifest.xml** 文件的默认内容：

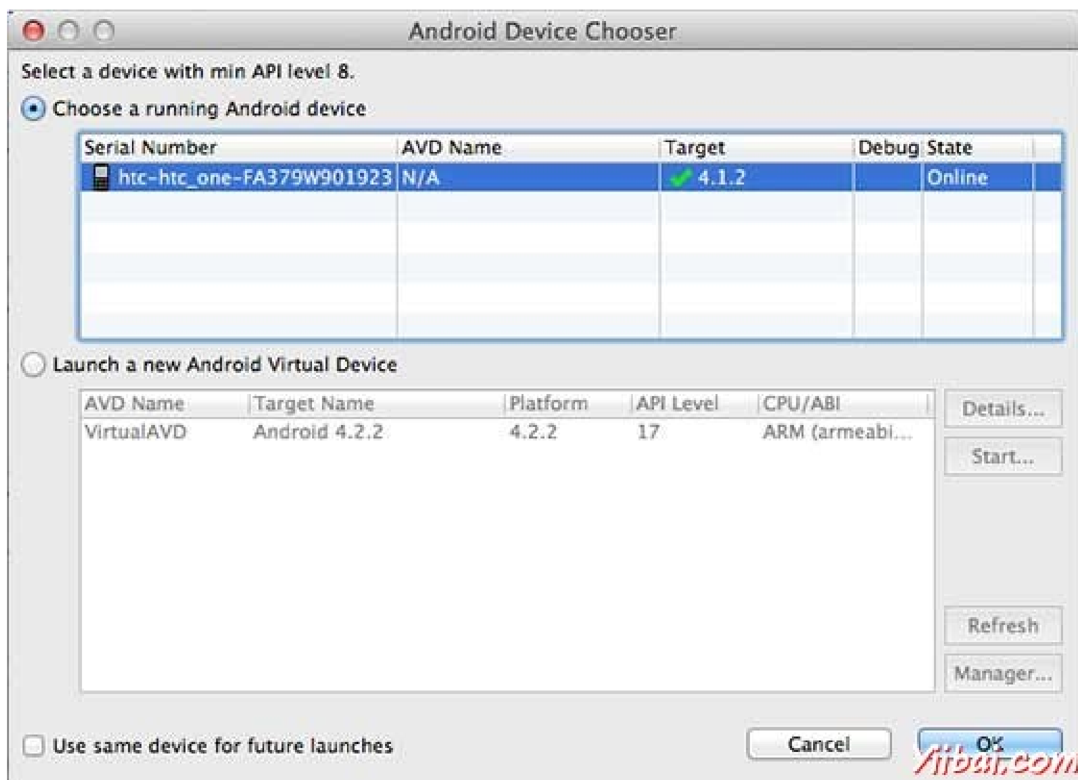
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/androic
    package="com.yiibai.sendsmsdemo"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

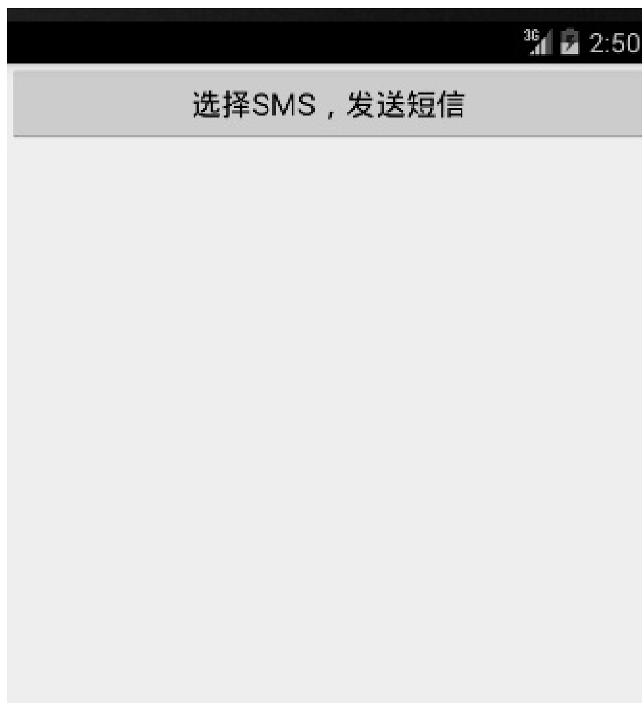
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.yiibai.sendsmsdemo.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" /

                <category android:name="android.intent.category.LAU
            </intent-filter>
        </activity>
    </application>
</manifest>
```

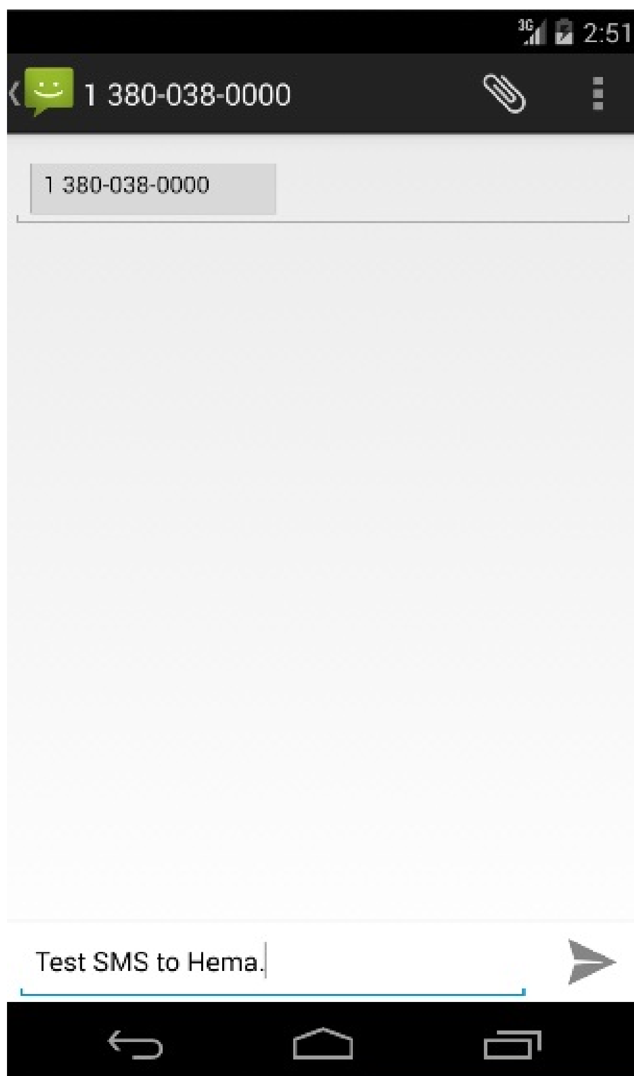
我们尝试运行 SendSMSDemo 应用程序。 Eclipse AVD安装的应用程序，并启动它，如果一切设置和应用都没有问题，它会显示以下模拟器窗口：



选择移动设备作为一个选项，然后检查移动设备，这将显示以下画面：



现在使用Compose SMS“按钮推出Android内置的SMS客户端，如下图所示：



可以修改默认字段最后使用发送短信按钮（标有红色矩形）提到收件人发送短信。
以上示例代码下载：http://pan.baidu.com/s/1c0Ah508

Android拨打电话 - Android开发教程

每一个Android设备特别是手机都提供一个拨打电话功能，但仍然需要编写一个应用程序，给用户一个选择使用硬编码的电话号码拨打电话。

本章列出了一个简单的步骤来创建一个应用程序，它可以用来拨打电话。使用Android 的 Intent 通过调用Android内置的电话通话功能。以下部分说明 Intent 对象的拨打电话功能 。

Intent 对象 - 操作拨打电话

使用 ACTION_CALL 动作触发Android设备内置电话功能。以下是简单的语法用来创建一个Intent 的 ACTION_CALL 动作

```
Intent phoneIntent = new Intent(Intent.ACTION_CALL);
```

可以使用 ACTION_DIAL 动作，而不是 ACTION_CALL，在这种情况下，在使用选项来修改硬编码的电话号码拨打电话之前，而不是直接调用的。

Intent 对象 - 数据/电话呼叫类型

这里给定电话为 13800138000 拨打一个电话，需要使用setData()方法指定URI为 tel：如下：

```
phoneIntent.setData(Uri.parse("tel:13800138000"));
```

要注意的一点是，拨打电话不需要任何额外的数据或数据类型指定。

示例

下面的示例演示如何在实际使用 Android Intent 打电话给定的手机号码。

> 要尝试这个例子中，需要实际配备了最新的 Android OS 移动设备，否则仿真器可能无法正常工作。

步骤	描述
1	使用Android Studio创建Android应用程序，并将它命名为 PhoneCallDemounder。创建这个项目，确保目标 SDK编译在 Android SDK 的最新版本或使用更高级别的API
2	修改 *src/MainActivity.java* 文件，并添加所需的代码，以拨打电话
3	修改所需的布局XML文件 *res/layout/activity_main.xml* 添加GUI组件。添加一个简单的按钮来拨打号码：13800138000
4	修改 *res/values/strings.xml* 定义所需的常数值
5	修改 AndroidManifest.xml 如下所示
6	运行该应用程序启动 Android模拟器并验证应用程序所做的修改结果

以下是修改主活动文件 src/com.yiibai.phonecalldemo/MainActivity.java 的内容如下：

```
package com.yiibai.phonecalldemo;

import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      Button startBtn = (Button) findViewById(R.id.makeCall);
      startBtn.setOnClickListener(new View.OnClickListener() {
         public void onClick(View view) {
         makeCall();
      }
   });

   }
   protected void makeCall() {
      Log.i("Make call", "");

      Intent phoneIntent = new Intent(Intent.ACTION_CALL);
      phoneIntent.setData(Uri.parse("tel:91-800-001-0101"));

      try {
         startActivity(phoneIntent);
         finish();
         Log.i("Finished making a call...", "");
      } catch (android.content.ActivityNotFoundException ex) {
         Toast.makeText(MainActivity.this,
         "Call faild, please try again later.", Toast.LENGTH_SHORT
      }
   }
   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      // Inflate the menu; this adds items to the action bar if it
      getMenuInflater().inflate(R.menu.main, menu);
      return true;
   }
}
```

下面是 **res/layout/activity_main.xml** 文件的内容：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/and
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <Button android:id="@+id/makeCall"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:text="@string/make_call"/>

</LinearLayout>
```

下面文件 res/values/strings.xml 的内容中定义两个新的常量：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">PhoneCallDemo</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="make_call">Call 91-800-001-0101</string>

</resources>
```

以下是**AndroidManifest.xml** 文件的默认内容：

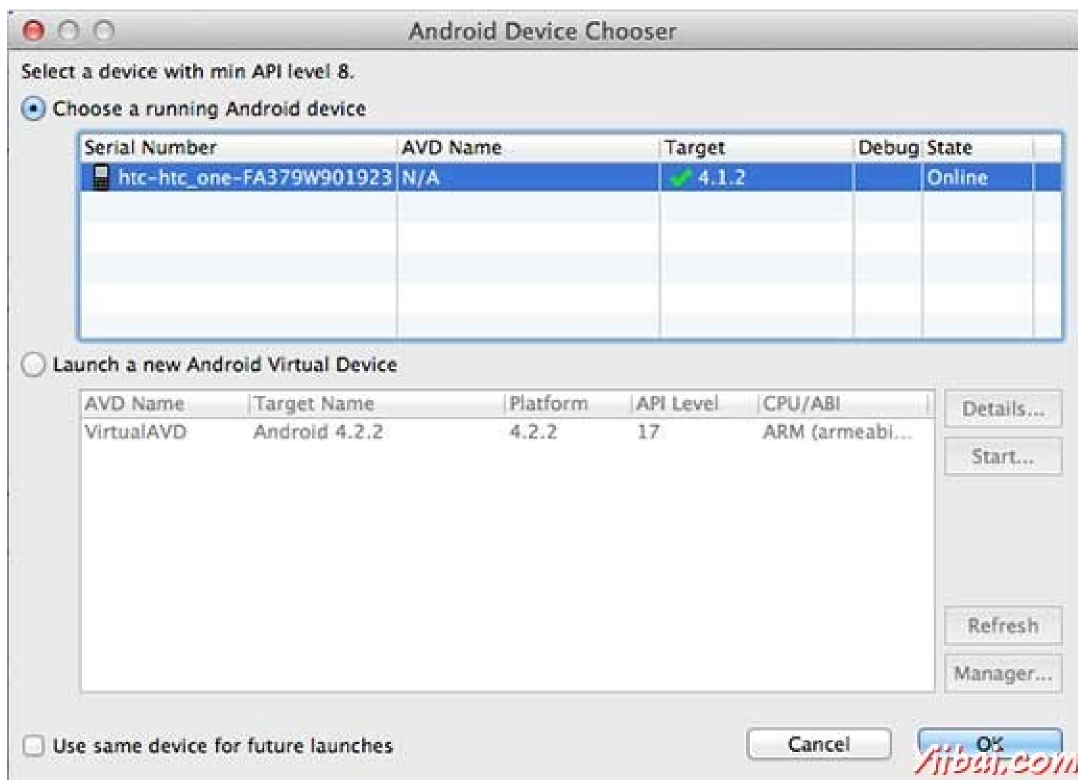
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/androic
    package="com.yiibai.phonecalldemo"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />
   <uses-permission android:name="android.permission.CALL_PHONE" />
   <uses-permission android:name="android.permission.READ_PHONE_STA

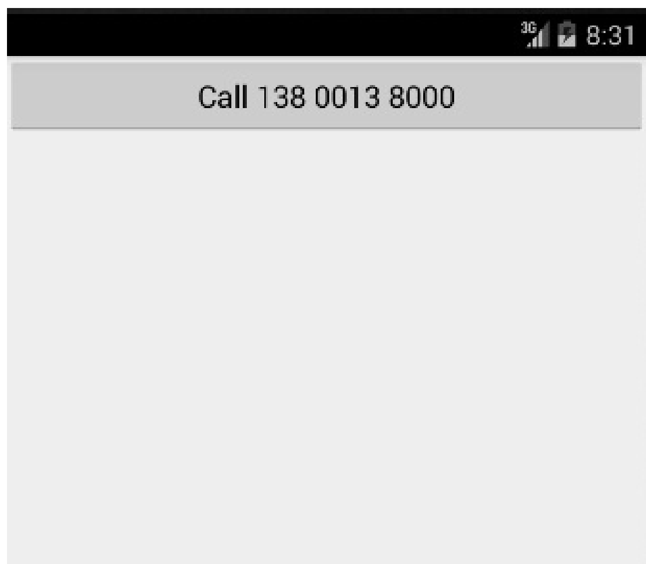
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.yiibai.phonecalldemo.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" /

                <category android:name="android.intent.category.LAU
            </intent-filter>
        </activity>
    </application>
</manifest>
```

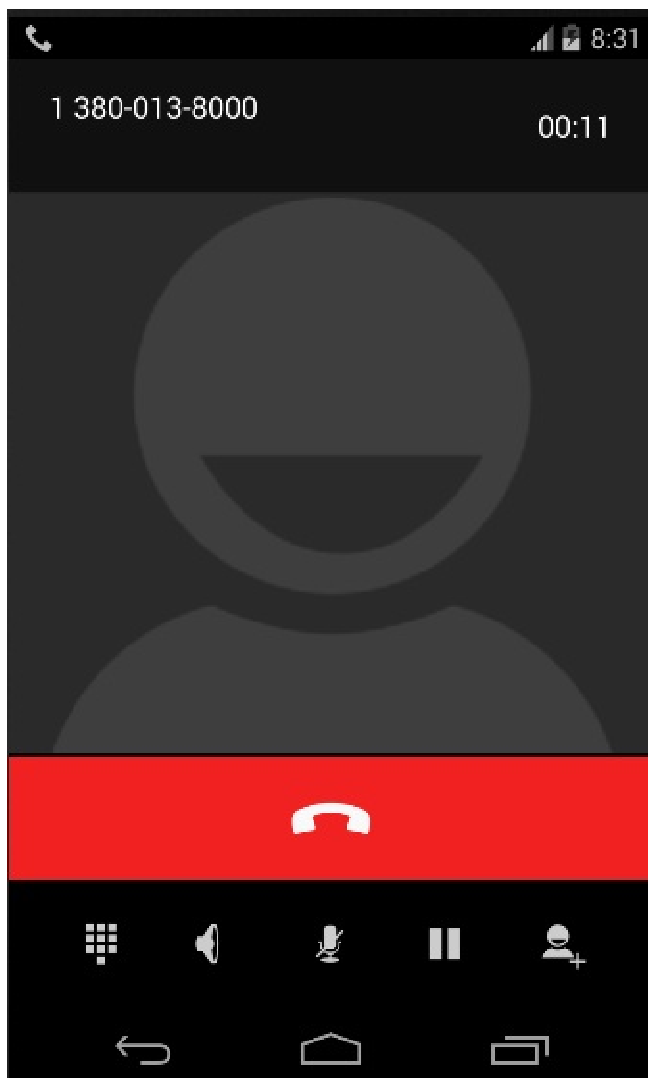
我们尝试运行PhoneCallDemo 应用程序。Eclipse AVD安装的应用程序，并启动它，如果一切设置和应用代码都没有问题，它会显示以下模拟器窗口：



选择移动设备作为一个选项，然后检查移动设备，这将显示以下画面：



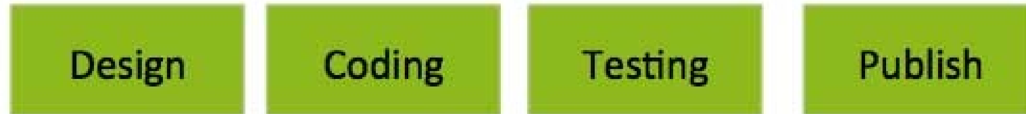
现在使用按钮拨打138001380000，如下所示：



以上代码下载：
http://pan.baidu.com/s/1hq1RSuK

发布Android应用 - Android开发教程

Android应用程序的发布是一个过程，让Android的应用程序提供给用户。发布的Android应用程序开发过程的最后阶段。



一旦开发和全面测试Android应用程序，就可以开始销售或分发免费使用谷歌播放（著名的Android市场）。也可以发布应用程序，通过它们直接发送给用户，让用户下载他们从自己的网站。

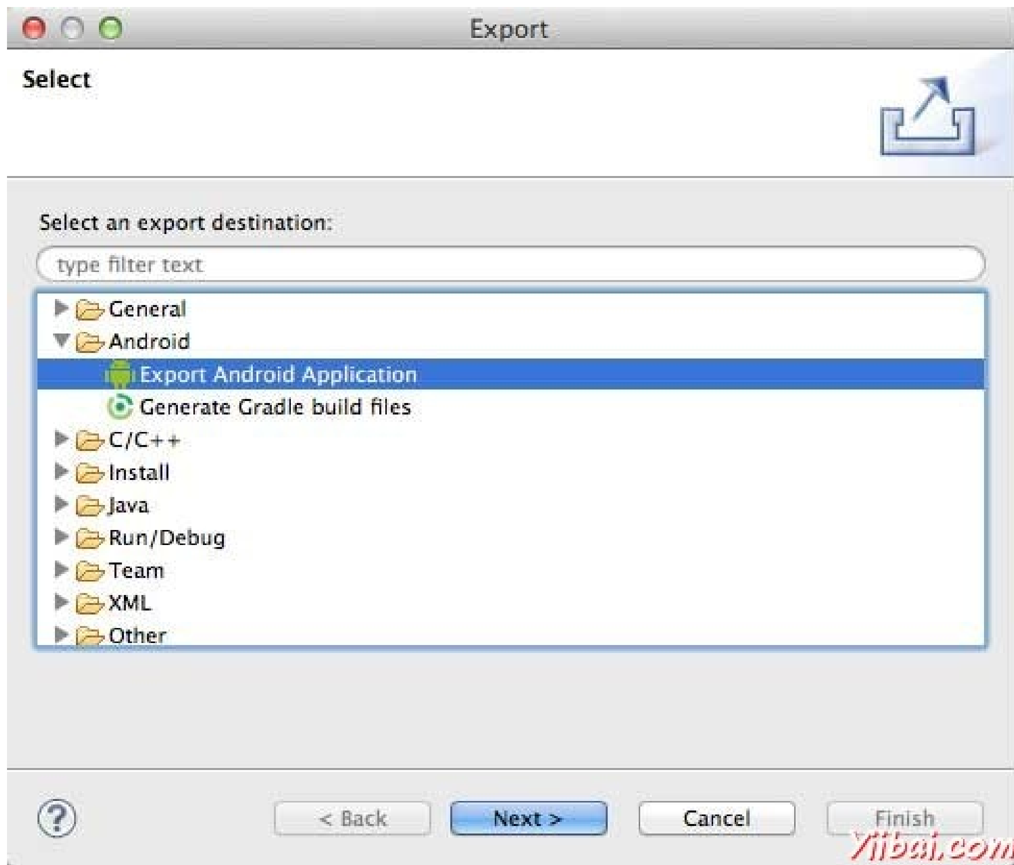
在Android官方网站上，可以检查详细的出版过程中，但本教程将通过简单的步骤来启动应用程序，谷歌播放。下面是一个简化的检查清单，这将帮助在推出Android应用程序：

步骤	Activity
1	**Regression Testing** Before you publish your application, you need to make sure that its meeting the basic quality expectations for all Android apps, on all of the devices that you are targeting. So perform all the required testing on different devices including phone and tablets.
2	**Application Rating** When you will publish your application at Google Play, you will have to specify a content rating for your app, which informs Google Play users of its maturity level. Currently available ratings are (a) Everyone (b) Low maturity (c) Medium maturity (d) High maturity.
3	**Targeted Regions** Google Play lets you control what countries and territories where your application will be sold. Accordingly you must take care of setting up time zone, localization or any other specific requirement as per the targeted region.
4	**Application Size** Currently, the maximum size for an APK published on Google Play is 50 MB. If your app exceeds that size, or if you want to offer a secondary download, you can use APK Expansion Files, which Google Play will host for free on its server infrastructure and automatically handle the download to devices.
5	**SDK and Screen Compatibility** It is important to make sure that your app is designed to run properly on the Android platform versions and device screen sizes that you want to target.
6	**Application Pricing** Deciding whether you app will be free or paid is important because, on Google Play, free apps must remain free. If you want to sell your application then you will have to specify its price in different currencies.
7	**Promotional Content** It is a good marketing practice to supply a variety of high-quality graphic assets to showcase your app or brand. After you publish, these appear on your product details page, in store listings and search results, and elsewhere.
8	**Build and Upload release-ready APK** The release-ready APK is what you you will upload to the Developer Console and distribute to users. You can check complete detail on how to create a release-ready version of your app: Preparing for Release.
9	**Finalize Application Detail** Google Play gives you a variety of ways to promote your app and engage with users on your product details page, from colorful graphics, screenshots, and videos to localized descriptions, release details, and links to your other apps. So you can decorate your application page and provide as much as clear crisp detail you can provide.

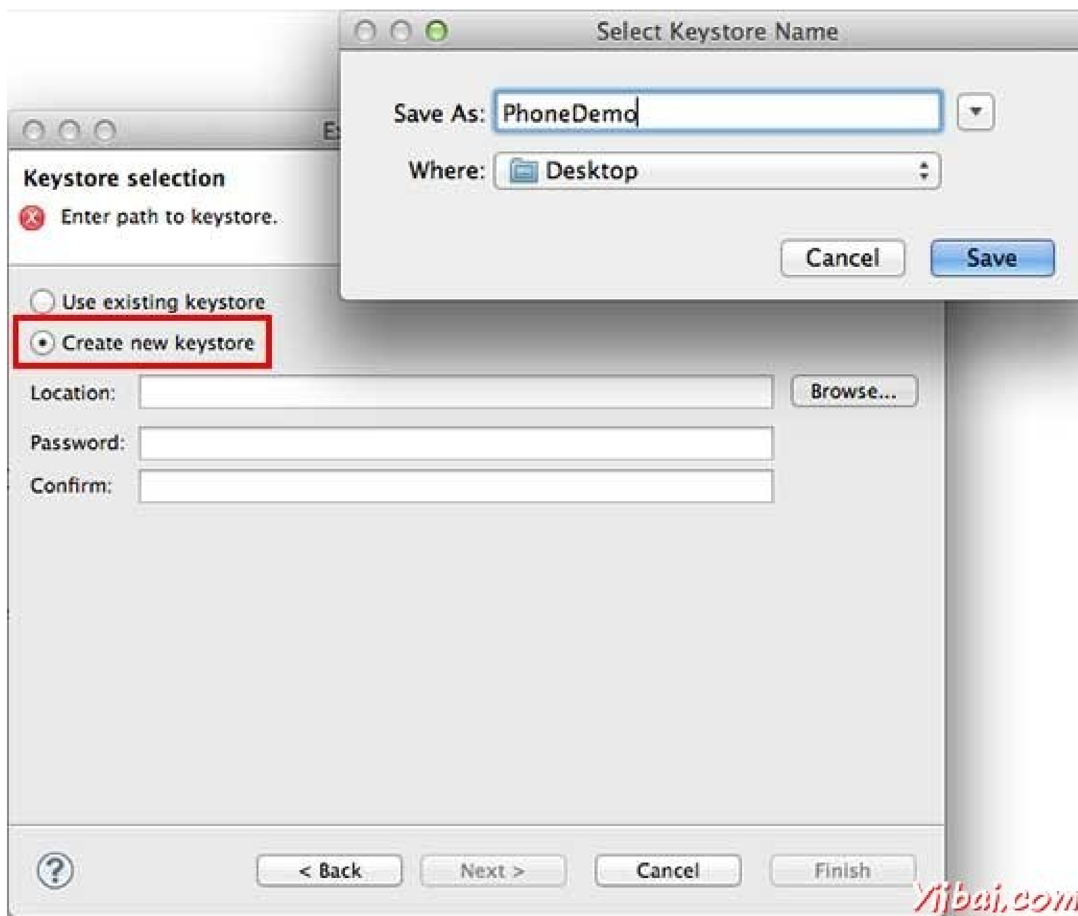
导出Android应用

作为一个APK（Android包）文件，需要将应用程序导出之前上传谷歌播放市场。

要导出应用程序，只要打开该应用程序的Eclipse项目中，从Eclipse中选择File->Export 并按照简单的步骤来导出应用程序：



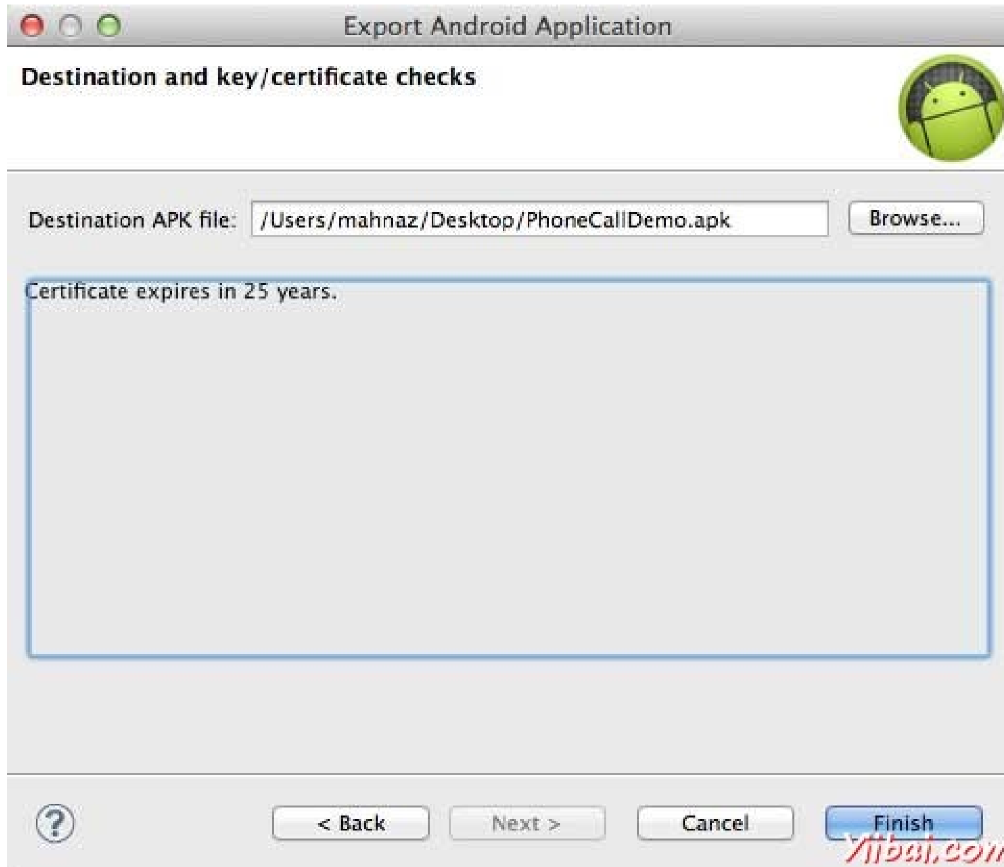
接下来选择 Export Android Application“选项，在上面的屏幕截图所示，然后单击”Next “，再 Next，这样得到的画面，在那里选择”Create new keystore" 来存储应用程序。



输入密码来保护应用程序，并单击“Next ”按钮，再次。它会显示以下画面，让应用程序创建一个密钥：



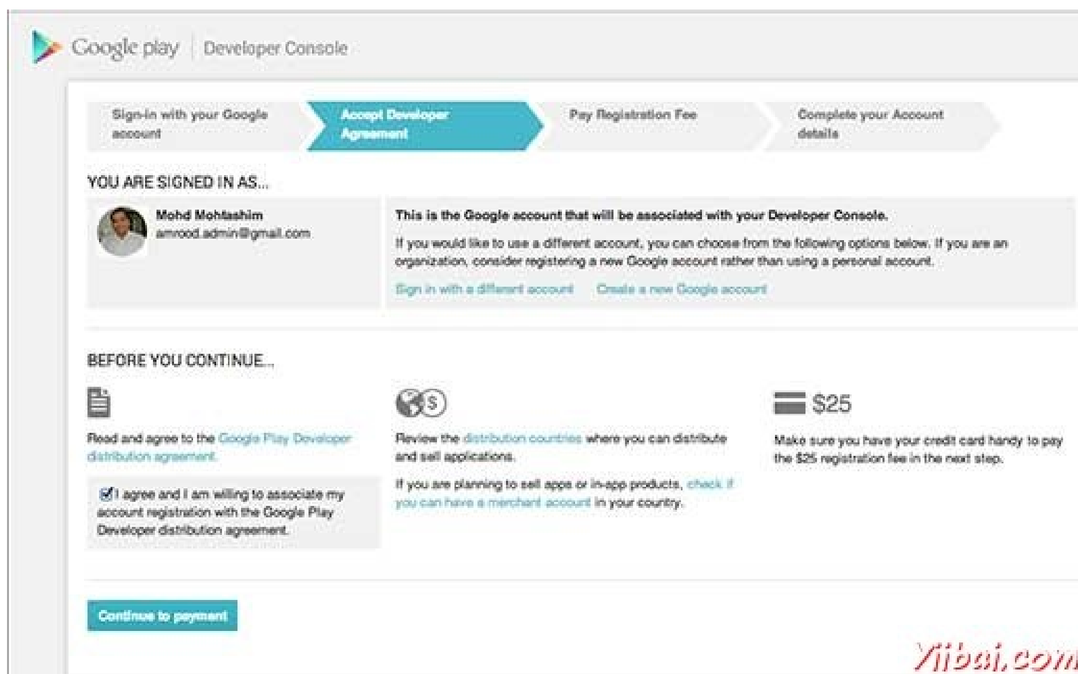
一旦填写的所有信息，单击“Next ”按钮，最后它会问申请将导出的位置：



最后，单击“Finish ”按钮，生成Android应用程序包文件将被上传在谷歌播放市场。

谷歌播放注册

最重要的一步是注册谷歌播放使用谷歌播放市场。可以使用您现有的Google ID，如果没有可以创建一个新的Google ID，然后注册与市场。将有以下屏幕接受的条款和条件。



可以使用Continue to payment“按钮，继续进行付款登记费$25，并最终完成帐户细节。

一旦你是一个注册用户在Google Play，您可以上传您的应用程序release-ready APK，最后将完成应用程序的详细使用上述清单的第9步中提到的应用程序的详细页面。

W3School Ionic 教程

作者：W3School

来源：Ionic 教程

ionic 入门

ionic 简介



ionic 是一个强大的 HTML5 应用程序开发框架(HTML5 Hybrid Mobile App Framework)。 可以帮助您使用 Web 技术，比如 HTML、CSS 和 Javascript 构建接近原生体验的移动应用程序。

ionic 主要关注外观和体验，以及和你的应用程序的 UI 交互，特别适合用于基于 Hybird 模式的 HTML5 移动应用程序开发。

ionic是一个轻量的手机UI库，具有速度快，界面现代化、美观等特点。为了解决其他一些UI库在手机上运行缓慢的问题，它直接放弃了IOS6和Android4.1以下的版本支持，来获取更好的使用体验。

ionic 特点

- 1.ionic 基于Angular语法，简单易学。
- 2.ionic 是一个轻量级框架。
- 3.ionic 完美的融合下一代移动框架，支持 Angularjs 的特性， MVC ，代码易维护。
- 4.ionic 提供了漂亮的设计，通过 SASS 构建应用程序，它提供了很多 UI 组件来帮助开发者开发强大的应用。
- 5.ionic 专注原生，让你看不出混合应用和原生的区别
- 6.ionic 提供了强大的命令行工具。
- 7.ionic 性能优越，运行速度快。

学习本教程前你需要了解？

学习本教程前你需要了解以下基础知识：

- HTML
- CSS
- Javascript
- Angular

ionic 相关内容

ionic 官方网站：http://ionicframework.com/

ionic 官方文档：http://ionicframework.com/docs/

Github 地址：https://github.com/driftyco/ionic

ionic 安装

本站实例采用了ionic v1.0.1 版本，下载地址为：ionic-v1.0.1.zip。

ionic 最新版本下载地址：http://ionicframework.com/docs/overview/#download。

下载后解压压缩包，包含以下目录：

```
css/                =>    样式文件
fonts/              =>    字体文件
js/                 =>    Javascript文件
version.json        =>    版本更新说明
```

你也可以在 Github 上下载以下资源文件：https://github.com/driftyco/ionic（在 release 目录中）。

接下来，我们只需要在项目中引入以上目录中的 css/ionic.min.css 和 js/ionic.bundle.min.js 文件即可创建 ionic 应用。

实例

```
<ion-header-bar class="bar-positive">
    <h1 class="title">Hello World!</h1>
</ion-header-bar>

<ion-content>
    <p>我的第一个 ionic 应用。</p>
</ion-content>
```

点击 "尝试一下" 按钮查看在线实例。

本教程着重讲解 ionic 框架的应用，大部分实例在浏览器中运行，移动设备上运行可以在接下来的命令行安装教程中详细了解。

注意：在移动应用如 phonegap 中我们只需将对应的 js 和 css 文件加入到资源库中即可。

命令行安装

首先您需要安装 Node.js，我们在接下来的安装中需要使用到其 NPM 工具，更多 NPM 介绍可以查看我们的NPM 使用介绍。

然后通过命令行工具安装最新版本的 cordova 和 ionic 。通过参考Android 和 iOS 官方文档来安装。

Window 和 Linux 上打开命令行工具执行以下命令：

```
$ npm install -g cordova ionic
```

Mac 系统上使用以下命令：

```
sudo npm install -g cordova ionic
```

提示*: IOS*需要在*Mac Os X.* 和*Xcode*环境下面安装使用。

如果你已经安装了以上环境，可以执行以下命令来更新版本:

```
npm update -g cordova ionic
```

或

```
sudo npm update -g cordova ionic
```

创建应用

使用ionic官方提供的现成的应用程序模板，或一个空白的项目创建一个ionic应用：

```
$ ionic start myApp tabs
```

运行我们刚才创建的**ionic**项目

使用 ionic tool 创建，测试，运行你的apps(或者通过Cordova直接创建)。

创建android应用:

```
$ cd myApp
$ ionic platform add android
$ ionic build android
$ ionic emulate android
```

创建ios应用:

```
$ cd myApp
$ ionic platform add ios
$ ionic build ios
$ ionic emulate ios
```

ionic 创建 APP

前面的章节中我们已经学会了 ionic 框架如何导入到项目中。

接下来我们将为大家介绍如何创建一个 ionic APP 应用。

ionic 创建 APP 使用 HTML、CSS 和 Javascript 来构建，所以我们可以创建一个 www 目录，并在目录下创建 index.html 文件，代码如下：

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Todo</title>
    <meta name="viewport" content="initial-scale=1, maximum-scale=1

    <link href="lib/ionic/css/ionic.css" rel="stylesheet">

    <script src="lib/ionic/js/ionic.bundle.js"></script>

    <!-- 在使用 Cordova/PhoneGap 创建的 APP 中包含的文件，由 Cordova/Ph
    <script src="cordova.js"></script>
  </head>
  <body>
  </body>
</html>
```

以上代码中，我们引入了 Ionic CSS 文件、 Ionic JS 文件及 Ionic AngularJS 扩展 ionic.bundle.js（ionic.bundle.js）。

ionic.bundle.js 文件已经包含了 Ionic核心JS、AngularJS、Ionic的AngularJS扩展，如果你需要引入其他 Angular 模块，可以从 lib/js/angular 目录中调用。

cordova.js 是在使用 Cordova/PhoneGap 创建应用时生成的，不需要手动引入，你可以在 Cordova/PhoneGap 项目中找到该文件，所以在开发过程中显示 404 是正常的。

创建 APP

接下来我们来实现一个包含标题、侧边栏、列表等的应用,设计图如下：



创建侧边栏

侧边栏创建使用 ion-side-menus 控制器。

编辑我们先前创建的 index.html 文件，修改 <body> 内的内容，如下所示：

```
<body>
  <ion-side-menus>
    <ion-side-menu-content>
    </ion-side-menu-content>
    <ion-side-menu side="left">
    </ion-side-menu>
  </ion-side-menus>
</body>
```

控制器解析：

- **ion-side-menus**： 是一个带有边栏菜单的容器，可以通过点击按钮或者滑动屏幕来展开菜单。

- **ion-side-menu-content**：展示主要内容的容器，可以通过滑动屏幕来展开 menu。
- **ion-side-menu**：存放侧边栏的容器。

初始化 APP

接下来我们创建一个新的 AngularJS 模块，并初始化，代码位于 www/js/app.js 中：

```
angular.module('todo', ['ionic'])
```

之后在我们的 body 标签中添加 ng-app 属性：

```
<body ng-app="todo">
```

在 index.html 文件的 <script src="cordova.js"></script> 上面引入 app.js 文件：

```
<script src="js/app.js"></script>
```

修改 index.html 文件 body 标签的内容，代码如下所示：

```
<body ng-app="todo">
  <ion-side-menus>

    <!-- 中心内容 -->
    <ion-side-menu-content>
      <ion-header-bar class="bar-dark">
        <h1 class="title">Todo</h1>
      </ion-header-bar>
      <ion-content>
      </ion-content>
    </ion-side-menu-content>

    <!-- 左侧菜单 -->
    <ion-side-menu side="left">
      <ion-header-bar class="bar-dark">
        <h1 class="title">Projects</h1>
      </ion-header-bar>
    </ion-side-menu>

  </ion-side-menus>
</body>
```

以上步骤我们已经完成了 ionic 基本 APP 的应用。

创建列表

首先创建一个控制器 **TodoCtrl**：

```
<body ng-app="todo" ng-controller="TodoCtrl">
```

在app.js文件中，使用控制器定义列表数据：

```
angular.module('todo', ['ionic'])

.controller('TodoCtrl', function($scope) {
  $scope.tasks = [
    { title: '菜鸟教程' },
    { title: 'www.runoob.com' },
    { title: '菜鸟教程' },
    { title: 'www.runoob.com' }
  ];
});
```

在index.html页面中数据列表我们使用 Angular ng-repeat 来迭代数据：

```
<!-- 中心内容 -->
<ion-side-menu-content>
  <ion-header-bar class="bar-dark">
    <h1 class="title">Todo</h1>
  </ion-header-bar>
  <ion-content>
    <!-- 列表 -->
    <ion-list>
      <ion-item ng-repeat="task in tasks">
        {{task.title}}
      </ion-item>
    </ion-list>
  </ion-content>
</ion-side-menu-content>
```

修改后 index.html body 标签内代码如下：

```
 <body ng-app="todo" ng-controller="TodoCtrl">
    <ion-side-menus>

    <!-- 中心内容 -->
    <ion-side-menu-content>
      <ion-header-bar class="bar-dark">
        <h1 class="title">Todo</h1>
      </ion-header-bar>
      <ion-content>
        <!-- 列表 -->
        <ion-list>
          <ion-item ng-repeat="task in tasks">
            {{task.title}}
          </ion-item>
        </ion-list>
      </ion-content>
    </ion-side-menu-content>

    <!-- 左侧菜单 -->
    <ion-side-menu side="left">
      <ion-header-bar class="bar-dark">
        <h1 class="title">Projects</h1>
      </ion-header-bar>
    </ion-side-menu>

    </ion-side-menus>
    <script src="http://www.runoob.com/static/ionic/js/app.js"></sc
    <script src="cordova.js"></script>
</body>
```

创建添加页面

修改 index.html 在 **</ion-side-menus>** 后添加如下代码:

```
<script id="new-task.html" type="text/ng-template">

  <div class="modal">

    <!-- Modal header bar -->
    <ion-header-bar class="bar-secondary">
      <h1 class="title">New Task</h1>
      <button class="button button-clear button-positive" ng-click=
    </ion-header-bar>

    <!-- Modal content area -->
    <ion-content>

      <form ng-submit="createTask(task)">
        <div class="list">
          <label class="item item-input">
            <input type="text" placeholder="What do you need to do?
          </label>
        </div>
        <div class="padding">
          <button type="submit" class="button button-block button-p
        </div>
      </form>

    </ion-content>

  </div>

</script>
```

以上代码中我们通过 **<script id="new-task.html" type="text/ng-template">** 定义了新的模板页面。

表单提交时触发 createTask(task) 函数。

ng-model="task.title" 会将表单的输入数据赋值给 task 对象的 title 属性。

修改 **<ion-side-menu-content>** 内的内容，代码如下：

```
<!-- 中心内容 -->
<ion-side-menu-content>
<ion-header-bar class="bar-dark">
  <h1 class="title">Todo</h1>
  <!-- 新增按钮-->
  <button class="button button-icon" ng-click="newTask()">
    <i class="icon ion-compose"></i>
  </button>
</ion-header-bar>
<ion-content>
  <!-- 列表 -->
  <ion-list>
    <ion-item ng-repeat="task in tasks">
      {{task.title}}
    </ion-item>
  </ion-list>
</ion-content>
</ion-side-menu-content>
```

app.js 文件中，控制器代码如下：

```
angular.module('todo', ['ionic'])

.controller('TodoCtrl', function($scope, $ionicModal) {
  $scope.tasks = [
    { title: '菜鸟教程' },
    { title: 'www.runoob.com' },
    { title: '菜鸟教程' },
    { title: 'www.runoob.com' }
  ];

  // 创建并载入模型
  $ionicModal.fromTemplateUrl('new-task.html', function(modal) {
    $scope.taskModal = modal;
  }, {
    scope: $scope,
    animation: 'slide-in-up'
  });

  // 表单提交时调用
  $scope.createTask = function(task) {
    $scope.tasks.push({
      title: task.title
    });
    $scope.taskModal.hide();
    task.title = "";
  };

  // 打开新增的模型
  $scope.newTask = function() {
    $scope.taskModal.show();
  };

  // 关闭新增的模型
  $scope.closeNewTask = function() {
    $scope.taskModal.hide();
  };
});
```

创建侧边栏

通过以上步骤我们已经实现了新增功能，现在我们为 app 添加侧边栏功能。

修改 内的内容，代码如下：

```
<!-- 中心内容 -->
<ion-side-menu-content>
  <ion-header-bar class="bar-dark">
    <button class="button button-icon" ng-click="toggleProjects()">
      <i class="icon ion-navicon"></i>
    </button>
    <h1 class="title">{{activeProject.title}}</h1>
    <!-- 新增按钮 -->
    <button class="button button-icon" ng-click="newTask()">
      <i class="icon ion-compose"></i>
    </button>
  </ion-header-bar>
  <ion-content scroll="false">
    <ion-list>
      <ion-item ng-repeat="task in activeProject.tasks">
        {{task.title}}
      </ion-item>
    </ion-list>
  </ion-content>
</ion-side-menu-content>
```

添加侧边栏：

```
<!-- 左边栏 -->
<ion-side-menu side="left">
<ion-header-bar class="bar-dark">
  <h1 class="title">Projects</h1>
  <button class="button button-icon ion-plus" ng-click="newProject(
  </button>
</ion-header-bar>
<ion-content scroll="false">
  <ion-list>
    <ion-item ng-repeat="project in projects" ng-click="selectProje
      {{project.title}}
    </ion-item>
  </ion-list>
</ion-content>
</ion-side-menu>
```

<ion-item> 中的 ng-class 指令设置菜单激活样式。

这里我们创建一个新的js 文件 app2.js(为了不与前面的混淆)，代码如下：

```
angular.module('todo', ['ionic'])
/**
 * The Projects factory handles saving and loading projects
 * from local storage, and also lets us save and load the
```

```
  * last active project index.
  */
 .factory('Projects', function() {
   return {
     all: function() {
       var projectString = window.localStorage['projects'];
       if(projectString) {
         return angular.fromJson(projectString);
       }
       return [];
     },
     save: function(projects) {
       window.localStorage['projects'] = angular.toJson(projects);
     },
     newProject: function(projectTitle) {
       // Add a new project
       return {
         title: projectTitle,
         tasks: []
       };
     },
     getLastActiveIndex: function() {
       return parseInt(window.localStorage['lastActiveProject']) ||
     },
     setLastActiveIndex: function(index) {
       window.localStorage['lastActiveProject'] = index;
     }
   }
 })

 .controller('TodoCtrl', function($scope, $timeout, $ionicModal, Pro

   // A utility function for creating a new project
   // with the given projectTitle
   var createProject = function(projectTitle) {
     var newProject = Projects.newProject(projectTitle);
     $scope.projects.push(newProject);
     Projects.save($scope.projects);
     $scope.selectProject(newProject, $scope.projects.length-1);
   }

   // Load or initialize projects
   $scope.projects = Projects.all();

   // Grab the last active, or the first project
   $scope.activeProject = $scope.projects[Projects.getLastActiveInde

   // Called to create a new project
   $scope.newProject = function() {
     var projectTitle = prompt('Project name');
     if(projectTitle) {
       createProject(projectTitle);
     }
```

```
  };

  // Called to select the given project
  $scope.selectProject = function(project, index) {
    $scope.activeProject = project;
    Projects.setLastActiveIndex(index);
    $ionicSideMenuDelegate.toggleLeft(false);
  };

  // Create our modal
  $ionicModal.fromTemplateUrl('new-task.html', function(modal) {
    $scope.taskModal = modal;
  }, {
    scope: $scope
  });

  $scope.createTask = function(task) {
    if(!$scope.activeProject || !task) {
      return;
    }
    $scope.activeProject.tasks.push({
      title: task.title
    });
    $scope.taskModal.hide();

    // Inefficient, but save all the projects
    Projects.save($scope.projects);

    task.title = "";
  };

  $scope.newTask = function() {
    $scope.taskModal.show();
  };

  $scope.closeNewTask = function() {
    $scope.taskModal.hide();
  }

  $scope.toggleProjects = function() {
    $ionicSideMenuDelegate.toggleLeft();
  };

  // Try to create the first project, make sure to defer
  // this by using $timeout so everything is initialized
  // properly
  $timeout(function() {
    if($scope.projects.length == 0) {
      while(true) {
        var projectTitle = prompt('Your first project title:');
        if(projectTitle) {
          createProject(projectTitle);
          break;
```

```
        }
      }
    }
  });

});
```

body 中 ion-side-menus 代码如下：：

```
<ion-side-menus>

<!-- 中心内容 -->
<ion-side-menu-content>
  <ion-header-bar class="bar-dark">
    <button class="button button-icon" ng-click="toggleProjects()">
      <i class="icon ion-navicon"></i>
    </button>
    <h1 class="title">{{activeProject.title}}</h1>
    <!-- 新增按钮 -->
    <button class="button button-icon" ng-click="newTask()">
      <i class="icon ion-compose"></i>
    </button>
  </ion-header-bar>
  <ion-content scroll="false">
    <ion-list>
      <ion-item ng-repeat="task in activeProject.tasks">
        {{task.title}}
      </ion-item>
    </ion-list>
  </ion-content>
</ion-side-menu-content>

<!-- 左边栏 -->
<ion-side-menu side="left">
<ion-header-bar class="bar-dark">
  <h1 class="title">Projects</h1>
  <button class="button button-icon ion-plus" ng-click="newProject(
  </button>
</ion-header-bar>
<ion-content scroll="false">
  <ion-list>
    <ion-item ng-repeat="project in projects" ng-click="selectProje
      {{project.title}}
    </ion-item>
  </ion-list>
</ion-content>
</ion-side-menu>

</ion-side-menus>
```

ionic CSS

ionic 头部与底部

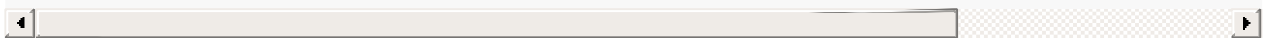
Header(头部)

Header是固定在屏幕顶部的组件,可以包如标题和左右的功能按钮。

ionic 默认提供了许多种颜色样式，你可以调用不同的样式名，当然也可以自定义一个。

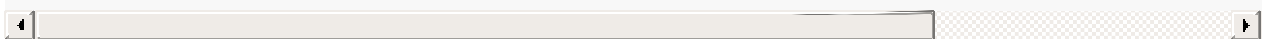
bar-light

```
<div  class="bar bar-header bar-light">  <h1  class="title">bar-lig
```



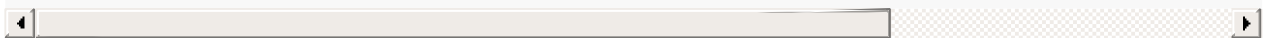
bar-stable

```
<div  class="bar bar-header bar-stable">  <h1  class="title">bar-st
```



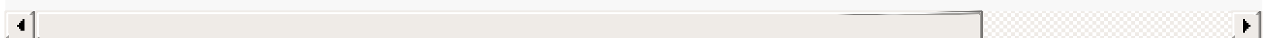
bar-positive

```
<div  class="bar bar-header bar-positive">  <h1  class="title">bar-
```



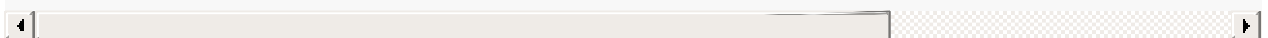
bar-calm

```
<div  class="bar bar-header bar-calm">  <h1  class="title">bar-calm
```



bar-balanced

```
<div  class="bar bar-header bar-balanced">  <h1  class="title">bar-
```



bar-energized

```
<div  class="bar bar-header bar-energized">  <h1  class="title">bar
```

bar-assertive

```
<div  class="bar bar-header bar-assertive">  <h1  class="title">bar
```

bar-royal

```
<div  class="bar bar-header bar-royal">  <h1  class="title">bar-roy
```

bar-dark

```
<div  class="bar bar-header bar-dark">  <h1  class="title">bar-dark
```

Sub Header（副标题）

Sub Header同样是固定在顶部，只是是在Header的下面，就算没有写Header这个，Sub Header这个样式也会距离顶部有一个Header的距离。颜色样式同 Header 。

```
<div  class="bar bar-header">  <h1  class="title">Header</h1>  </di
```

Footer(底部)

Footer 是在屏幕的最下方，可以包含多种内容类型。

```
<div  class="bar bar-footer bar-balanced">  <div  class="title">Foo
```

Footer 同上面的 Header，只是把样式名 bar-header 换做 bar-footer 。

```
<div  class="bar bar-footer">  <button  class="button button-clear'
```

此外，如果底部没有标题，但是又需要右边的按钮，你需要在右侧按钮添加 pull-right如:

```
<div  class="bar bar-footer">  <button  class="button button-clear
```

ionic 按钮

按钮是移动app不可或缺的一部分，不同风格的app，需要的不同按钮的样式。

默认情况下，按钮显示样式为：**display: inline-block**。

```
<button class="button">
  Default
</button>

<button class="button button-light">
  button-light
</button>

<button class="button button-stable">
  button-stable
</button>

<button class="button button-positive">
  button-positive
</button>

<button class="button button-calm">
  button-calm
</button>

<button class="button button-balanced">
  button-balanced
</button>

<button class="button button-energized">
  button-energized
</button>

<button class="button button-assertive">
  button-assertive
</button>

<button class="button button-royal">
  button-royal
</button>

<button class="button button-dark">
  button-dark
</button>
```

button-block 样式按钮显示为：**display: block**，它将完全填充父元素的宽度，包含了内边距属性padding。

```
<button class="button button-block button-positive">
  Block Button
</button>
```

使用 button-full 类，可以让按钮显示完全宽度，且不包含内边距padding。

```
<button class="button button-full button-positive">
  Full Width Block Button
</button>
```

不同大小的按钮

button-large 设置为大按钮，button-small 设置为小按钮。

```
<button class="button button-small button-assertive">
  Small Button
</button>
<button class="button button-large button-positive">
  Large Button
</button>
```

无背景按钮

button-outline 设置背景为透明。

```
<button class="button button-outline button-positive">
  Outlined Button
</button>
```

无背景与边框按钮

button-clear 设置按钮背景为透明，且无边框。

```
<button class="button button-clear button-positive">
  Clear Button
</button>
```

图标按钮

我们可以在按钮上添加图标。

```
<button class="button">
  <i class="icon ion-loading-c"></i> Loading...
</button>

<button class="button icon-left ion-home">Home</button>

<button class="button icon-left ion-star button-positive">Favorites

<a class="button icon-right ion-chevron-right button-calm">Learn Mo

<a class="button icon-left ion-chevron-left button-clear button-dar

<button class="button icon ion-gear-a"></button>

<a class="button button-icon icon ion-settings"></a>

<a class="button button-outline icon-right ion-navicon button-balar
```

头部/底部添加按钮

头部/底部可以添加按钮，按钮的样式根据头部/底部来设定，所以你不需要为按钮添加额外的样式。

```
<div class="bar bar-header">
  <button class="button icon ion-navicon"></button>
  <h1 class="title">Header Buttons</h1>
  <button class="button">Edit</button>
</div>
```

button-clear 类可以设置无背景和边框的头部/底部按钮。

```
<div class="bar bar-header">
  <button class="button button-icon icon ion-navicon"></button>
  <div class="h1 title">Header Buttons</div>
  <button class="button button-clear button-positive">Edit</button>
</div>
```

按钮栏

我们可以使用 button-bar 类来设置按钮栏。以下实例中，我们在头部和内容中添加了按钮栏。

```
<div class="button-bar">
  <a class="button">First</a>
  <a class="button">Second</a>
  <a class="button">Third</a>
</div>
```

ionic 列表

列表是一个应用广泛的界面元素，在所有移动app中几乎都会使用到。

列表可以是基本文字、按钮，开关，图标和缩略图等。

列表项可以是任何的HTML元素。容器元素需要list类，每个列表项需要使用item类。

ionList和ionItem可以很容易的支持各种交互方式，比如，滑动编辑，拖动排序，以及删除项。

基本用法:

```
<ul class="list">
    <li class="item">
      ...
    </li>
</ul>
```

列表分隔符

我们可以使用 item-divider 类来为列表创建分隔符，默认情况下，列表项以不同的背景颜色和字体加粗来区分，但你也可以很容易的定制他。

```
<div class="list">

  <div class="item item-divider">
    Candy Bars
  </div>

  <a class="item" href="#">
    Butterfinger
  </a>

  ...

</div>
```

带图标列表

我们可以在列表项的左侧或右侧指定图标。

使用 item-icon-left 图标在左侧， item-icon-right 设置图标在右侧。如果你需要在两边都有图标，则两个类都添加上即可。

以下实例中，我们在列表项中使用了 <a> 标签，使得每个列表项可点击。

列表项在使用 <a>或<button> 元素时，如果右侧未添加图标，则会自动添加上箭头号。

实例中，第一项只有左侧图标，第二项左右均有图标，第三项有右侧图标（还有注释 item-note），第四项有badge（标记）元素。

```
<div class="list">

  <a class="item item-icon-left" href="#">
    <i class="icon ion-email"></i>
    Check mail
  </a>

  <a class="item item-icon-left item-icon-right" href="#">
    <i class="icon ion-chatbubble-working"></i>
    Call Ma
    <i class="icon ion-ios-telephone-outline"></i>
  </a>

  <a class="item item-icon-left" href="#">
    <i class="icon ion-mic-a"></i>
    Record album
    <span class="item-note">
      Grammy
    </span>
  </a>

  <a class="item item-icon-left" href="#">
    <i class="icon ion-person-stalker"></i>
    Friends
    <span class="badge badge-assertive">0</span>
  </a>

</div>
```

按钮列表

使用 item-button-right 或 item-button-left 类将按钮放在列表项中。

```
<div class="list">

  <div class="item item-button-right">
    Call Ma
    <button class="button button-positive">
      <i class="icon ion-ios-telephone"></i>
    </button>
  </div>

  ...

</div>
```

带头像列表

使用 item-avatar 来创建一个带头像的列表：

```
<div class="list">

    <a class="item item-avatar" href="#">
      <img src="venkman.jpg">
      <h2>Venkman</h2>
      <p>Back off, man. I'm a scientist.</p>
    </a>

    ...

</div>
```

缩略图列表

item-thumbnail-left 类用于添加左侧对齐的缩略图， item-thumbnail-right 类用于添加右侧对齐的缩略图。

```
<div class="list">

    <a class="item item-thumbnail-left" href="#">
      <img src="cover.jpg">
      <h2>Pretty Hate Machine</h2>
      <p>Nine Inch Nails</p>
    </a>

    ...

</div>
```

内嵌列表(inset list)

我们可以在容器当中内嵌列表，列表不会显示完整的宽度。

内嵌列表的样式为：list list-inset，与常规列表区别是，它设置了外边距（marign）,类似于选项卡。

内嵌列表是没有阴影效果的，滚动时效果会更好。

```
<div class="list list-inset">

    <div class="item">
      Raiders of the Lost Ark
    </div>

    ...

</div>
```

ionic 卡片

近年来卡片(card)的应用越来越流行，卡片提供了一个更好组织信息展示的工具。

针对移动端的应用，卡片会根据屏幕大小自适应大小。

我们可以很灵活的控制卡片的显示效果，甚至实现动画效果。

卡片一般放在页面顶部，当然也可以实现左右切换的功能。

```
<div class="card">
  <div class="item item-text-wrap">
    基本卡片，包含了文本信息。
  </div>
</div>
```

卡片(card)默认样式带有box-shadow(阴影)，由于性能的原因，和他类似的元素像 list list-inset 并没有阴影。

如果你有很多的卡片，每个卡片都有很多子元素，建议使用内嵌列表（inset list）。

卡片的头部与底部

我们可以通过添加 item-divider 类为卡片添加头部与底部：

```
<div class="card">
  <div class="item item-divider">
    卡片头部！
  </div>
  <div class="item item-text-wrap">
    基本卡片，包含了文本信息。
  </div>
  <div class="item item-divider">
    卡片底部！
  </div>
</div>
```

卡片列表

使用 list card 类来设置卡片列表：

```
<div class="list card">

  <a href="#" class="item item-icon-left">
    <i class="icon ion-home"></i>
    Enter home address
  </a>

  <a href="#" class="item item-icon-left">
    <i class="icon ion-ios-telephone"></i>
    Enter phone number
  </a>

  <a href="#" class="item item-icon-left">
    <i class="icon ion-wifi"></i>
    Enter wireless password
  </a>

  <a href="#" class="item item-icon-left">
    <i class="icon ion-card"></i>
    Enter card information
  </a>

</div>
```

带图片卡片

卡片中使用图片，效果会更好，实例如下：

```
<div class="list card">

  <div class="item item-avatar">
    <img src="avatar.jpg">
    <h2>Pretty Hate Machine</h2>
    <p>Nine Inch Nails</p>
  </div>

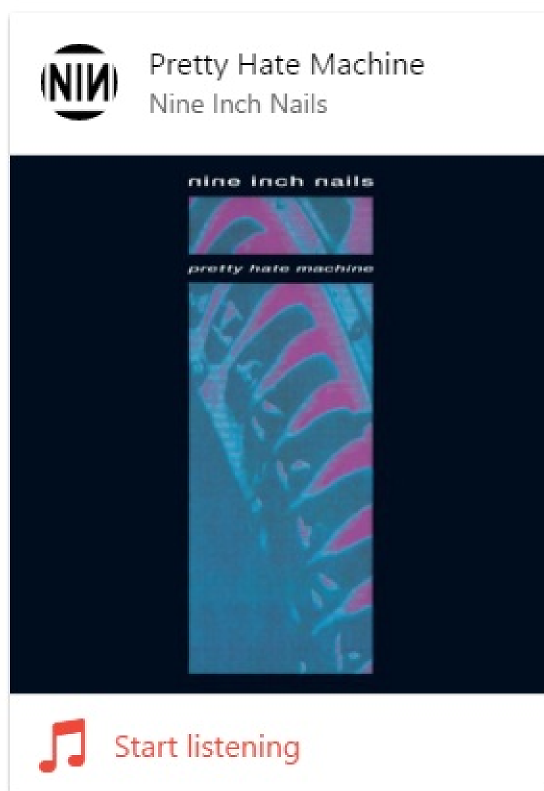
  <div class="item item-image">
    <img src="cover.jpg">
  </div>

  <a class="item item-icon-left assertive" href="#">
    <i class="icon ion-music-note"></i>
    Start listening
  </a>

</div>
```

运行效果如下：

卡片



卡片展现

以下实例中使用几种不同的选项的卡片展现方式。 开始使用了 list card 元素，并使用了 item-avatar , item-body 元素用于展示图片和文本信息，底部使用 item-divider 类。

```
<div class="list card">

  <div class="item item-avatar">
    <img src="mcfly.jpg">
    <h2>Marty McFly</h2>
    <p>November 05, 1955</p>
  </div>

  <div class="item item-body">
    <img class="full-image" src="delorean.jpg">
    <p>
      菜鸟教程 -- 学的不仅是技术，更新梦想！<br>
      菜鸟教程 -- 学的不仅是技术，更新梦想！<br>
      菜鸟教程 -- 学的不仅是技术，更新梦想！<br>
      菜鸟教程 -- 学的不仅是技术，更新梦想！
    </p>
    <p>
      <a href="#" class="subdued">1 喜欢</a>
      <a href="#" class="subdued">5 评论</a>
    </p>
  </div>

  <div class="item tabs tabs-secondary tabs-icon-left">
    <a class="tab-item" href="#">
      <i class="icon ion-thumbsup"></i>
      喜欢
    </a>
    <a class="tab-item" href="#">
      <i class="icon ion-chatbox"></i>
      Comment
    </a>
    <a class="tab-item" href="#">
      <i class="icon ion-share"></i>
      分享
    </a>
  </div>

</div>
```

运行效果如下：

卡片

Marty McFly
November 05, 1955

菜鸟教程 -- 学的不仅是技术，更新梦想！
菜鸟教程 -- 学的不仅是技术，更新梦想！
菜鸟教程 -- 学的不仅是技术，更新梦想！
菜鸟教程 -- 学的不仅是技术，更新梦想！

1 喜欢 5 评论

喜欢 评论 分享

ionic 表单和输入框

list 类同样可以用于 input 元素。item-input 和 item 类指定了文本框及其标签。

输入框属性：placeholder

以下实例中，默认为100%宽度（左右两侧没有边框），并使用 placeholder 属性设置输入字段预期值的提示信息。

```
<div class="list">
  <label class="item item-input">
    <input type="text" placeholder="First Name">
  </label>
  <label class="item item-input">
    <input type="text" placeholder="Last Name">
  </label>
  <label class="item item-input">
    <textarea placeholder="Comments"></textarea>
  </label>
</div>
```

输入框属性：input-label

使用 input-label 将标签放置于输入框 input 的左侧。

```
<div class="list">
  <label class="item item-input">
    <span class="input-label">用户名：</span>
    <input type="text">
  </label>
  <label class="item item-input">
    <span class="input-label">密码：</span>
    <input type="password">
  </label>
</div>
```

堆叠标签

堆叠标签通常位于输入框的头部。每个选项使用 item-stacked-label 类指定。 每个输入框需要指定 input-label。以下实例也使用了 placeholder 属性来设置信息输入提示。

```
<div class="list">
  <label class="item item-input item-stacked-label">
    <span class="input-label">First Name</span>
    <input type="text" placeholder="John">
  </label>
  <label class="item item-input item-stacked-label">
    <span class="input-label">Last Name</span>
    <input type="text" placeholder="Suhr">
  </label>
  <label class="item item-input item-stacked-label">
    <span class="input-label">Email</span>
    <input type="text" placeholder="john@suhr.com">
  </label>
</div>
```

浮动标签

浮动标签类似于堆叠标签，但浮动标签有一个动画的效果，每个选项需要指定 item-floating-label 类， 输入标签需要指定 input-label。

```
<div class="list">
  <label class="item item-input item-floating-label">
    <span class="input-label">First Name</span>
    <input type="text" placeholder="First Name">
  </label>
  <label class="item item-input item-floating-label">
    <span class="input-label">Last Name</span>
    <input type="text" placeholder="Last Name">
  </label>
  <label class="item item-input item-floating-label">
    <span class="input-label">Email</span>
    <input type="text" placeholder="Email">
  </label>
</div>
```

内嵌表单

默认情况下每个输入域宽度都是100%，但我们可以使用 list list-inset 或 card 类设置表单的内边距(padding)， card 类带有阴影。

```
<div class="list list-inset">
  <label class="item item-input">
    <input type="text" placeholder="First Name">
  </label>
  <label class="item item-input">
    <input type="text" placeholder="Last Name">
  </label>
</div>
```

内嵌输入域

使用 list-inset 设置内嵌实体列表。 使用 item-input-inset 样式可以内嵌一个按钮。

```
<div class="list">

  <div class="item item-input-inset">
    <label class="item-input-wrapper">
      <input type="text" placeholder="Email">
    </label>
    <button class="button button-small">
      Submit
    </button>
  </div>

</div>
```

带图标的输入框

item-input 输入框可以很简单的添加图标。 图标可以在 <input> 前添加。

```
<div class="list list-inset">
  <label class="item item-input">
    <i class="icon ion-search placeholder-icon"></i>
    <input type="text" placeholder="Search">
  </label>
</div>
```

头部输入框

输入框可放置在头部，并可添加提交或取消按钮。

```
<div class="bar bar-header item-input-inset">
  <label class="item-input-wrapper">
    <i class="icon ion-ios-search placeholder-icon"></i>
    <input type="search" placeholder="搜索">
  </label>
  <button class="button button-clear">
    取消
  </button>
</div>
```

ionic Toggle(切换开关)

切换开关类似与 HTML 的 checkbox 标签，但它更易于在移动设备上使用。

切换开关可以使用 toggle-assertive 来指定颜色。

```
<label class="toggle">
   <input type="checkbox">
   <div class="track">
     <div class="handle"></div>
   </div>
</label>
```

该实例有是多个切换开关列表。注意，每个选项的 item 类后需要添加 item-toggle 类。

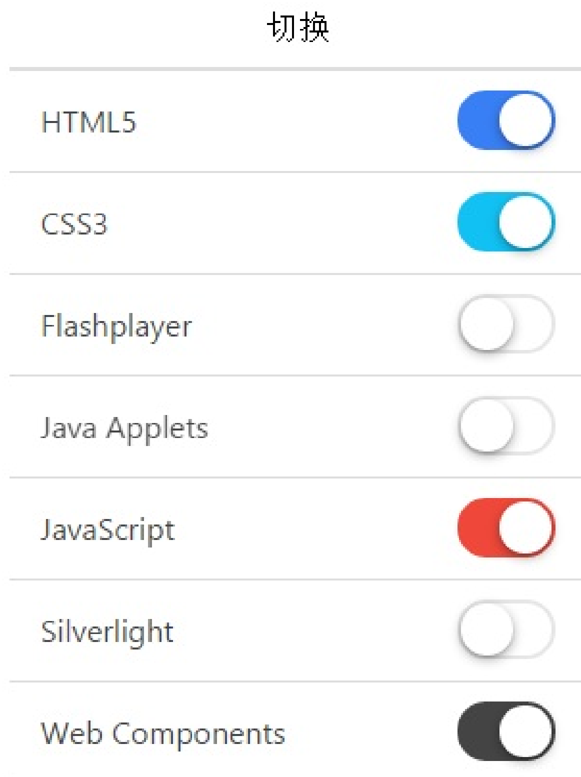
```
<ul class="list">

  <li class="item item-toggle">
     HTML5
     <label class="toggle toggle-assertive">
       <input type="checkbox">
       <div class="track">
         <div class="handle"></div>
       </div>
     </label>
  </li>

  ...

</ul>
```

运行效果如下：



ionic checkbox（复选框）

ionic 里面的 Checkbox 和普通的 Checkbox 效果上其实相差不大，只是样式上有所不同。

以下实例颜色了多个复选框的列表。

注意，每个选项的 item 类后需要添加 item-checkbox 类。

复选框可以使用 checkbox-assertive 来指定颜色。

```
<ul class="list">

  <li class="item item-checkbox">
     <label class="checkbox">
       <input type="checkbox">
     </label>
     Flux Capacitor
  </li>

  ...

</ul>
```

运行效果如下：

复选框

- [x] Flux Capacitor
- [x] 1.21 Gigawatts
- [x] Delorean
- [x] 88 MPH
- [ ] Plutonium Resupply

ionic 单选框

ionic 当选按钮与标准 type="radio" 的 input元素类似。以下展示了现代app类型的单选按钮。

每个 item-radio 中的 type="radio" 的 input 元素的 name 属性都相同。radio-icon 类用于是否显示图标。

ionic 在单选项中使用了 <label> 元素，使其更易点击。

实例

```
<div class="list">

<label class="item item-radio">
  <input type="radio" name="group" value="go" checked="checked">
  <div class="item-content">
    Go
  </div>
  <i class="radio-icon ion-checkmark"></i>
</label>

<label class="item item-radio">
  <input type="radio" name="group" value="python">
  <div class="item-content">
    Python
  </div>
  <i class="radio-icon ion-checkmark"></i>
</label>

<label class="item item-radio">
  <input type="radio" name="group" value="ruby">
  <div class="item-content">
    Ruby
  </div>
  <i class="radio-icon ion-checkmark"></i>
</label>

<label class="item item-radio">
  <input type="radio" name="group" value=".net">
  <div class="item-content">
    .Net
  </div>
  <i class="radio-icon ion-checkmark"></i>
</label>

<label class="item item-radio">
  <input type="radio" name="group" value="java">
```

```
    <div class="item-content">
      Java
    </div>
    <i class="radio-icon ion-checkmark"></i>
  </label>

  <label class="item item-radio">
    <input type="radio" name="group" value="php">
    <div class="item-content">
      PHP
    </div>
    <i class="radio-icon ion-checkmark"></i>
  </label>

  </div>
```

运行效果如下：

单选按钮列表

Go

Python

Ruby

.Net

Java

PHP

ionic Range

ionic Range 是一个滑块控件，ionic 为 Range 提供了很多种默认的样式。而且你可以在许多种元素里使用它比如列表或者 Card 。

实例

```
<div class="range">
  <i class="icon ion-volume-low"></i>
  <input type="range" name="volume">
  <i class="icon ion-volume-high"></i>
</div>

<div class="list" style="margin-top: 13px">
  <div class="item item-divider">
    Ranges In A List
  </div>
  <div class="item range range-positive">
    <i class="icon ion-ios-sunny-outline"></i>
    <input type="range" name="volume" min="0" max="100" value="12">
    <i class="icon ion-ios-sunny"></i>
  </div>
  <div class="item range range-calm">
    <i class="icon ion-ios-lightbulb-outline"></i>
    <input type="range" name="volume" min="0" max="100" value="25">
    <i class="icon ion-ios-lightbulb"></i>
  </div>
  <div class="item range range-balanced">
    <i class="icon ion-ios-bolt-outline"></i>
    <input type="range" name="volume" min="0" max="100" value="38">
    <i class="icon ion-ios-bolt"></i>
  </div>
  <div class="item range range-energized">
    <i class="icon ion-ios-moon-outline"></i>
    <input type="range" name="volume" min="0" max="100" value="50">
    <i class="icon ion-ios-moon"></i>
  </div>
  <div class="item range range-assertive">
    <i class="icon ion-ios-partlysunny-outline"></i>
    <input type="range" name="volume" min="0" max="100" value="63">
    <i class="icon ion-ios-partlysunny"></i>
  </div>
  <div class="item range range-royal">
    <i class="icon ion-ios-rainy-outline"></i>
    <input type="range" name="volume" min="0" max="100" value="75">
    <i class="icon ion-ios-rainy"></i>
  </div>
  <div class="item range range-dark">
    <i class="icon ion-ios-lightbulb-outline"></i>
    <input type="range" name="volume" min="0" max="100" value="88">
    <i class="icon ion-ios-lightbulb"></i>
  </div>
</div>
```

运行效果如下：

Range（滑块控件）

Ranges In A List

ionic select

ionic select 的 select 相比原生的要更加美观些。但是弹出的可选选项样式是浏览器默认的。

每个平台上的可选项样式都是不一样的，在PC电脑的浏览器上，你会看到传统的下拉界面，Android 上会弹出单选按钮选项，iOS 有个滚轮操作界面。

实例

```
<div  class="list">  <div  class="item item-input item-select">  <d
```

运行效果如下：

Select

Lightsaber	Green
Fighter	X-wing
Droid	R2-D2
Planet	Dagobah

ionic tab(选项卡)

ionic tab(选项卡) 是水平排列的按钮或者链接，用以页面间导航的切换。它可以包含文字和图标的组合，是一种移动设备上流行的导航方法。

以下选项卡容器使用了 tabs 类，每个选项卡使用 tab-item 类。默认情况下，选项卡是文本的，并没有图标。

实例

```
<div class="tabs">
  <a class="tab-item">
    主页
  </a>
  <a class="tab-item">
    收藏
  </a>
  <a class="tab-item">
    设置
  </a>
</div>
```

默认情况，选项卡颜色为默认，你可以设置以下不同颜色样式：tabs-default，tabs-light， tabs-stable， tabs-positive， tabs-calm， tabs-balanced， tabs-energized， tabs-assertive， tabs-royal， tabs-dark。

要隐藏选项卡栏，可使用 tabs-item-hide 类。

图标选项卡

在 tabs 类后添加 tabs-icon-only 类可设置只显示图标选项卡。

```
<div class="tabs tabs-icon-only">
  <a class="tab-item">
    <i class="icon ion-home"></i>
  </a>
  <a class="tab-item">
    <i class="icon ion-star"></i>
  </a>
  <a class="tab-item">
    <i class="icon ion-gear-a"></i>
  </a>
</div>
```

顶部图标+文字选项卡

在 tabs 类后添加 ttabs-icon-top 类可设置顶部图标+文字选项卡。

```
<div class="tabs tabs-icon-top">
  <a class="tab-item" href="#">
    <i class="icon ion-home"></i>
    主页
  </a>
  <a class="tab-item" href="#">
    <i class="icon ion-star"></i>
    收藏
  </a>
  <a class="tab-item" href="#">
    <i class="icon ion-gear-a"></i>
    设置
  </a>
</div>
```

左侧图标+文字选项卡

在 tabs 类后添加 ttabs-icon-left 类可设置左侧图标+文字选项卡。

```
<div class="tabs tabs-icon-left">
  <a class="tab-item">
    <i class="icon ion-home"></i>
    主页
  </a>
  <a class="tab-item">
    <i class="icon ion-star"></i>
    收藏
  </a>
  <a class="tab-item">
    <i class="icon ion-gear-a"></i>
    设置
  </a>
</div>
```

条纹样式选项卡

可以在带有 tabs 的样式名的元素上添加 tabs-striped 来实现像 Android 风格的 tabs。也可以添加 tabs-top 来实现选项卡在页面顶部。

条纹选项卡颜色可通过 tabs-background-{color} 和 tabs-color-{color} 来控制，{color} 值可以是：default, light, stable, positive, calm, balanced, energized, assertive, royal, 或 dark。

注意：如果要再选项卡上设置头部标题，需要使用 has-tabs-top 类。

```
<div class="tabs-striped tabs-top tabs-background-positive tabs-col
    <div class="tabs">
      <a class="tab-item active" href="#">
        <i class="icon ion-home"></i>
        Test
      </a>
      <a class="tab-item" href="#">
        <i class="icon ion-star"></i>
        Favorites
      </a>
      <a class="tab-item" href="#">
        <i class="icon ion-gear-a"></i>
        Settings
      </a>
    </div>
  </div>
  <div class="tabs-striped tabs-color-assertive">
    <div class="tabs">
      <a class="tab-item active" href="#">
        <i class="icon ion-home"></i>
        Test
      </a>
      <a class="tab-item" href="#">
        <i class="icon ion-star"></i>
        Favorites
      </a>
      <a class="tab-item" href="#">
        <i class="icon ion-gear-a"></i>
        Settings
      </a>
    </div>
  </div>
```

运行效果如下：


条纹样式选项卡



ionic 网格(Grid)

ionic 的网格(Grid)和其他大部分框架有所不同，它采用了弹性盒子模型(Flexible Box Model)。而且在移动端，基本上的手机都支持。row 样式指定行，col 样式指定列。

同等大小网格

在带有 row 的样式的元素里如果包含了 col 的样式，col 就会设置为同等大小。

以下实例中 row 的样式包含了 5 个 col 样式，每个 col 的宽度为 20%。

```
<div class="row">
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
</div>
```

指定列宽

你可以设定一行中各个列的大小不一样。默认情况下，列都会被划分为同等大小。但你也可以按百分比来设置列的宽度（一行为 12 个网格）。

```
<div class="row">
  <div class="col col-50">.col.col-50</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
</div>

<div class="row">
  <div class="col col-75">.col.col-75</div>
  <div class="col">.col</div>
</div>

<div class="row">
  <div class="col">.col</div>
  <div class="col col-75">.col.col-75</div>
</div>

<div class="row">
  <div class="col">.col</div>
  <div class="col">.col</div>
</div>
```

注意：实例中，每个 col 样式会自动添加上边框和灰色背景。

下面列出了指定列宽的一些百分比的样式名:

.col-10	10%
.col-20	20%
.col-25	25%
.col-33	33.3333%
.col-50	50%
.col-67	66.6666%
.col-75	75%
.col-80	80%
.col-90	90%

有偏移量的网格

列可以设置左侧偏移量，实例如下：

```
<div class="row">
  <div class="col col-33 col-offset-33">.col</div>
  <div class="col">.col</div>
</div>

<div class="row">
  <div class="col col-33">.col</div>
  <div class="col col-33 col-offset-33">.col</div>
</div>

<div class="row">
  <div class="col col-33 col-offset-67">.col</div>
</div>
```

下面是一些百分比的偏移量样式名：

.col-offset-10	**10%**
.col-offset-20	20%
.col-offset-25	25%
.col-offset-33	33.3333%
.col-offset-50	50%
.col-offset-67	66.6666%
.col-offset-75	75%
.col-offset-80	80%
.col-offset-90	90%

纵向对齐网格

弹性盒子模型可以很容易设置列纵向对齐。纵向对齐包含顶部，中间部分，底部，可以应用到每一行的列，或者指定的某列。

实例中，最后一列设置了最高的内容用于更好的演示纵向对齐网格。

```
<div class="row">
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">1<br>2<br>3<br>4</div>
</div>

<div class="row">
  <div class="col col-top">.col</div>
  <div class="col col-center">.col</div>
  <div class="col col-bottom">.col</div>
  <div class="col">1<br>2<br>3<br>4</div>
</div>

<div class="row row-top">
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">1<br>2<br>3<br>4</div>
</div>

<div class="row row-center">
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">1<br>2<br>3<br>4</div>
</div>

<div class="row row-bottom">
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">1<br>2<br>3<br>4</div>
</div>
```

响应式网格

手持设备屏幕在切换时，例如横屏，竖屏等。就需要设置每行的网格可以实现根据不同宽度自适应大小。

不同设备响应式类的样式如下:

响应式类	描述
.responsive-sm	小于手机横屏
.responsive-md	小于平板竖屏
.responsive-lg	小于平板横屏

```
<div class="row responsive-sm">
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
  <div class="col">.col</div>
</div>
```

ionic 颜色

ionic 提供了很多颜色的配置，当然你可以根据自己的需要自定义颜色。

```
<ul class="list color-list-demo">
    <li class="item dark">
      light
      <span class="color-demo light-bg light-border"></span>
    </li>
    <li class="item stable-dark">
      stable
      <span class="color-demo stable-bg stable-border"></span>
    </li>
    <li class="item positive">
      positive
      <span class="color-demo positive-bg positive-border"></span>
    </li>
    <li class="item calm">
      calm
      <span class="color-demo calm-bg calm-border"></span>
    </li>
    <li class="item balanced">
      balanced
      <span class="color-demo balanced-bg balanced-border"></span>
    </li>
    <li class="item energized">
      energized
      <span class="color-demo energized-bg energized-border"></spar
    </li>
    <li class="item assertive">
      assertive
      <span class="color-demo assertive-bg assertive-border"></spar
    </li>
    <li class="item royal">
      royal
      <span class="color-demo royal-bg royal-border"></span>
    </li>
    <li class="item dark">
      dark
      <span class="color-demo dark-bg dark-border"></span>
    </li>
</ul>
```

实例运行结果：

颜色列表
light
stable
positive
calm
balanced
energized
assertive
royal
dark

ionic icon(图标)

ionic 也默认提供了许多的图标，大概有500多个。用法也非常的简单：

```
<i  class="icon icon ion-star"></i>
```

图标样式CDN地址：http://www.runoob.com/static/ionic/css/ionicons.min.css。

图标列表如下：

-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-

-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-

ionic JavaScript

ionic 上拉菜单(ActionSheet)

上拉菜单(ActionSheet)通过往上弹出的框，来让用户选择选项。

非常危险的选项会以高亮的红色来让人第一时间识别。你可以通过点击取消按钮或者点击空白的地方来让它消失。

实例

HTML 代码

```
<body ng-app="starter" ng-controller="actionsheetCtl" >

    <ion-pane>
        <ion-content >
            <h2 ng-click="show()">Action Sheet</h2>
        </ion-content>
    </ion-pane>
</body>
```

JavaScript 代码

在代码中触发上拉菜单，需要在你的 angular 控制器中使用 $ionicActionSheet 服务：

```
angular.module('starter', ['ionic'])

.run(function($ionicPlatform) {
  $ionicPlatform.ready(function() {
    // Hide the accessory bar by default (remove this to show the a
    // for form inputs)
    if(window.cordova && window.cordova.plugins.Keyboard) {
      cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);
    }
    if(window.StatusBar) {
      StatusBar.styleDefault();
    }
  });
})

.controller( 'actionsheetCtl',['$scope','$ionicActionSheet','$timeo
    $scope.show = function() {

        var hideSheet = $ionicActionSheet.show({
            buttons: [
              { text: '<b>Share</b> This' },
              { text: 'Move' }
            ],
            destructiveText: 'Delete',
            titleText: 'Modify your album',
            cancelText: 'Cancel',
            cancel: function() {
                 // add cancel code..
               },
            buttonClicked: function(index) {
              return true;
            }
        });

        $timeout(function() {
            hideSheet();
        }, 2000);

    };
}])
```

运行效果如下图：



ionic 背景层

我们经常需要在 UI 上，例如在弹出框、加载框、其他弹出层中显示或隐藏背景层。

在组件中可以使用$ionicBackdrop.retain()来显示背景层，使用$ionicBackdrop.release()隐藏背景层。

每次调用retain后，背景会一直显示，直到调用release消除背景层。

实例

HTML 代码

```
<body ng-app="starter" ng-controller="actionsheetCtl" >
    <ion-pane>
        <ion-content >
            <h2 ng-click="action()">$ionicBackdrop</h2>
        </ion-content>
    </ion-pane>
</body>
```

JavaScript 代码

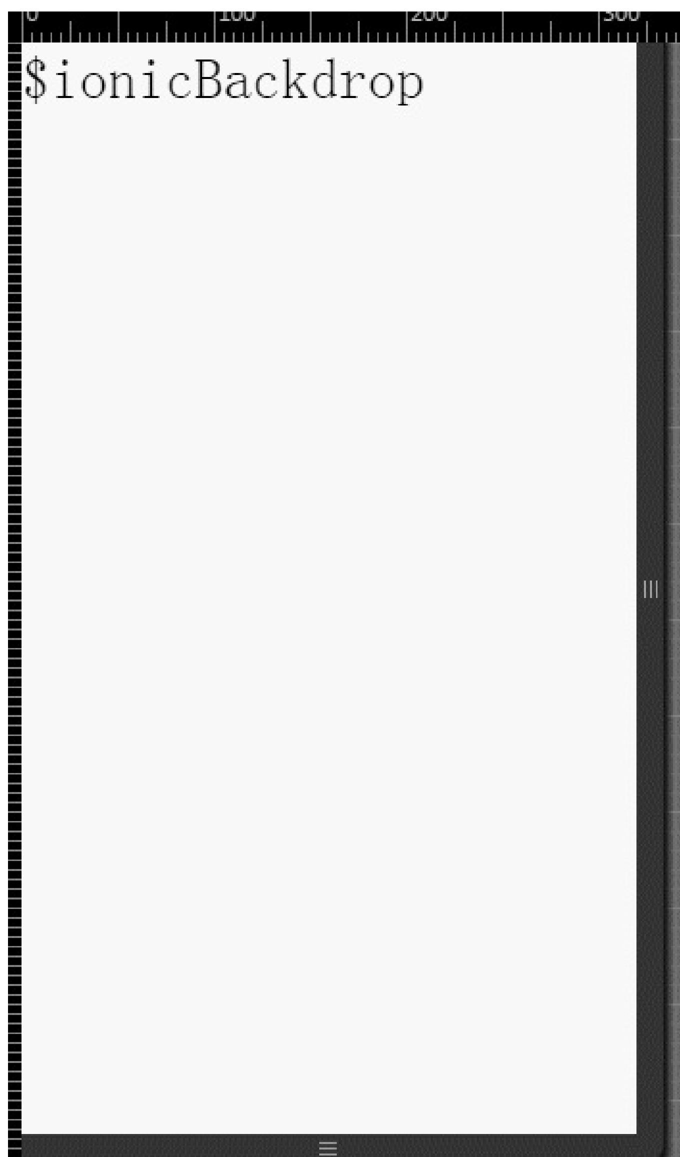
```
angular.module('starter', ['ionic'])

.run(function($ionicPlatform) {
  $ionicPlatform.ready(function() {
    // Hide the accessory bar by default (remove this to show the a
    // for form inputs)
    if(window.cordova && window.cordova.plugins.Keyboard) {
      cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);
    }
    if(window.StatusBar) {
      StatusBar.styleDefault();
    }
  });
})

.controller( 'actionsheetCtl',['$scope','$timeout' ,'$ionicBackdrop

    $scope.action = function() {
       $ionicBackdrop.retain();
       $timeout(function() {    //默认让它1秒后消失
         $ionicBackdrop.release();
       }, 1000);
    };
}])
```

显示效果如下图所示：


$ionicBackdrop

ionic 下拉刷新

在加载新数据的时候，我们需要实现下拉刷新效果，代码如下：

实例

HTML 代码

```
<body ng-app="starter" ng-controller="actionsheetCtl" >
    <ion-pane>
        <ion-content >
            <ion-refresher pulling-text="下拉刷新" on-refresh="doRef
            <ion-list>
                <ion-item ng-repeat="item in items" ng-bind="item.r
            </ion-list>
        </ion-content>
    </ion-pane>
</body>
```

JavaScript 代码

```
angular.module('starter', ['ionic'])

.run(function($ionicPlatform) {
  $ionicPlatform.ready(function() {
    // Hide the accessory bar by default (remove this to show the a
    // for form inputs)
    if(window.cordova && window.cordova.plugins.Keyboard) {
      cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);
    }
    if(window.StatusBar) {
      StatusBar.styleDefault();
    }
  });
})

.controller( 'actionsheetCtl',['$scope','$timeout' ,'$http',functio

    $scope.items=[
        {
            "name":"HTML5"
        },
        {
            "name":"JavaScript"
        },
        {
            "name":"Css3"
        }
    ];

    $scope.doRefresh = function() {
        $http.get('http://www.runoob.com/try/demo_source/item.json
            .success(function(newItems) {
                $scope.items = newItems;
            })
            .finally(function() {
                $scope.$broadcast('scroll.refreshComplete');
            });
    };
}])
```

item.json 文件数据：

```
[
    {
        "name":"菜鸟教程"
    },
    {
        "name":"www.runoob.com"
    }
]
```

效果如下所示：

HTML5

JavaScript

Css3

ionic 复选框

ionic 复选框（checkbox）与普通的 HTML 复选框没什么区别，以下实例演示了 ionic 复选框 ion-checkbox 的应用。

```
<ion-checkbox ng-model="isChecked">复选框标签</ion-checkbox>
```

实例

实例中，会根据复选框是否选中，修改 checked 值，true 为选中， false 为未选中。

HTML 代码

```
<ion-header-bar class="bar-positive">
  <h1 class="title">复选框</h1>
</ion-header-bar>

<ion-content>

  <div class="list">

    <ion-checkbox ng-repeat="item in devList"
                  ng-model="item.checked"
                  ng-checked="item.checked">
      {{ item.text }}
    </ion-checkbox>

    <div class="item">
      <div ng-bind="devList | json"></div>
    </div>

    <div class="item item-divider">
      Notifications
    </div>

    <ion-checkbox ng-model="pushNotification.checked"
                  ng-change="pushNotificationChange()">
      Push Notifications
    </ion-checkbox>

    <div class="item">
      <div ng-bind="pushNotification | json"></div>
    </div>

    <ion-checkbox ng-model="emailNotification"
                  ng-true-value="'Subscribed'"
                  ng-false-value="'Unubscribed'">
      Newsletter
    </ion-checkbox>
    <div class="item">
      <div ng-bind="emailNotification | json"></div>
    </div>

  </div>

</ion-content>
```

JavaScript 代码

```
angular.module('starter', ['ionic'])

.run(function($ionicPlatform) {
$ionicPlatform.ready(function() {
  // Hide the accessory bar by default (remove this to show the acc
  // for form inputs)
  if(window.cordova && window.cordova.plugins.Keyboard) {
    cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);
  }
  if(window.StatusBar) {
    StatusBar.styleDefault();
  }
});
})

.controller( 'actionsheetCtl',['$scope',function($scope){

  $scope.devList = [
      { text: "HTML5", checked: true },
      { text: "CSS3", checked: false },
      { text: "JavaScript", checked: false }
    ];

    $scope.pushNotificationChange = function() {
      console.log('Push Notification Change', $scope.pushNotificati
    };

    $scope.pushNotification = { checked: true };
    $scope.emailNotification = 'Subscribed';

}])
```

css 代码：

```
body {
  cursor: url('http://www.runoob.com/try/demo_source/finger.png'),
}
```

效果如下所示：

复选框

 HTML5

 CSS3

 JavaScript

```
[
  {
    "text": "HTML5",
    "checked": true
  },
  {
    "text": "CSS3",
    "checked": false
  }
```

ionic 单选框操作

实例中，根据选中的不同选项，显示不同的值。

HTML 代码

```
<ion-header-bar class="bar-positive">
  <h1 class="title">当选按钮</h1>
</ion-header-bar>

<ion-content>

  <div class="list">

    <div class="item item-divider">
      选取的值为: {{ data.clientSide }}
    </div>

    <ion-radio ng-repeat="item in clientSideList"
               ng-value="item.value"
               ng-model="data.clientSide">
      {{ item.text }}
    </ion-radio>

    <div class="item item-divider">
      Serverside, Selected Value: {{ data.serverSide }}
    </div>

    <ion-radio ng-repeat="item in serverSideList"
               ng-value="item.value"
               ng-model="data.serverSide"
               ng-change="serverSideChange(item)"
               name="server-side">
      {{ item.text }}
    </ion-radio>

  </div>

</ion-content>
```

JavaScript 代码

```
angular.module('ionicApp', ['ionic'])

.controller('MainCtrl', function($scope) {

  $scope.clientSideList = [
    { text: "Backbone", value: "bb" },
    { text: "Angular", value: "ng" },
    { text: "Ember", value: "em" },
    { text: "Knockout", value: "ko" }
  ];

  $scope.serverSideList = [
    { text: "Go", value: "go" },
    { text: "Python", value: "py" },
    { text: "Ruby", value: "rb" },
    { text: "Java", value: "jv" }
  ];

  $scope.data = {
    clientSide: 'ng'
  };

  $scope.serverSideChange = function(item) {
    console.log("Selected Serverside, text:", item.text, "value:",
  };

});
```

css 代码：

```
body {
  cursor: url('http://www.runoob.com/try/demo_source/finger.png'),
}
```

效果如下所示：

当选按钮

Backbone

Angular ✔

Ember

Knockout

Serverside, Selected Value:

Go

Python

ionic 切换开关操作

以下实例中，通过切换不同开关 checked 显示不同的值，true 为打开，false 为关闭。

HTML 代码

```
<ion-header-bar class="bar-positive">
  <h1 class="title">开关切换</h1>
</ion-header-bar>

<ion-content>

  <div class="list">

    <div class="item item-divider">
      Settings
    </div>

    <ion-toggle ng-repeat="item in settingsList"
                ng-model="item.checked"
                ng-checked="item.checked">
      {{ item.text }}
    </ion-toggle>

    <div class="item">
        <!-- 使用 pre 标签展示效果更美观 -->
      <div ng-bind="settingsList | json"></div>
    </div>

    <div class="item item-divider">
      Notifications
    </div>

    <ion-toggle ng-model="pushNotification.checked"
                ng-change="pushNotificationChange()">
      Push Notifications
    </ion-toggle>

    <div class="item">
        <!-- 使用 pre 标签展示效果更美观 -->
      <div ng-bind="pushNotification | json"></div>
    </div>

    <ion-toggle toggle-class="toggle-assertive"
                ng-model="emailNotification"
                ng-true-value="Subscribed"
                ng-false-value="Unubscribed">
```

```
      Newsletter
    </ion-toggle>

    <div class="item">
        <!-- 使用 pre 标签展示效果更美观 -->
      <div ng-bind="emailNotification | json"></div>
    </div>

  </div>

</ion-content>
```

由于pre标签冲突，实例中的 pre 已替换为 div标签，具体可以在"尝试一下"中查看。

JavaScript 代码

```
angular.module('ionicApp', ['ionic'])

.controller('MainCtrl', function($scope) {

  $scope.settingsList = [
    { text: "Wireless", checked: true },
    { text: "GPS", checked: false },
    { text: "Bluetooth", checked: false }
  ];

  $scope.pushNotificationChange = function() {
    console.log('Push Notification Change', $scope.pushNotificatio
  };

  $scope.pushNotification = { checked: true };
  $scope.emailNotification = 'Subscribed';

});
```

css 代码：

```
body {
  cursor: url('http://www.runoob.com/try/demo_source/finger.png'),
}
```

效果如下所示：

切换开关

Settings

Wireless

GPS

Bluetooth


```
[
  {
    "text": "Wireless",
    "checked": true
  },
  {
    "text": "GPS",
```

ionic 手势事件

`on-hold` ：长按的时间是500毫秒。

```
&lt;button
    on-hold="onHold()"
    class="button"&gt;
    Test
    &lt;/button&gt;
```

`on-tap` ：这个是手势轻击事件，如果长按时间超过250毫秒，那就不是轻击了。。

```
&lt;button
    on-tap="onTap()"
    class="button"&gt;
    Test
    &lt;/button&gt;
```

`on-double-tap` ：手双击屏幕事件

```
&lt;button
    on-double-tap="onDoubleTap()"
    class="button"&gt;
    Test
    &lt;/button&gt;
```

`on-touch` ：这个和 on-tap 还是有区别的，这个是立即执行，而且是用户点击立马执行。不用等待 touchend/mouseup 。

```
    &lt;button on-touch="onTouch()"
    class="button"&gt;
    Test
    &lt;/button&gt;
```

`on-release` ：当用户结束触摸事件时触发。

```
&lt;button
    on-release="onRelease()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-drag` ：这个有点类似于PC端的拖拽。当你一直点击某个物体，并且手开始移动，都会触发 on-drag。

```
&lt;button
    on-drag="onDrag()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-drag-up` ：向上拖拽。

```
&lt;button
    on-drag-up="onDragUp()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-drag-right` ：向右拖拽。

```
&lt;button
    on-drag-right="onDragRight()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-drag-down` ：向下拖拽。

```
&lt;button
    on-drag-down="onDragDown()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-drag-left` ：向左边拖拽。

```
&lt;button
    on-drag-left="onDragLeft()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-swipe` ：指手指滑动效果，可以是任何方向上的。而且也和 on-drag 类似，都有四个方向上单独的事件。

```
&lt;button
    on-swipe="onSwipe()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-swipe-up` ：向上的手指滑动效果。

```
&lt;button
    on-swipe-up="onSwipeUp()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-swipe-right` ：向右的手指滑动效果。

```
&lt;button
    on-swipe-right="onSwipeRight()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-swipe-down` ：向下的手指滑动效果。

```
&lt;button
    on-swipe-down="onSwipeDown()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

`on-swipe-left` ：向左的手指滑动效果。

```
&lt;button
    on-swipe-left="onSwipeLeft()"
    class="button"&gt;
    Test
&lt;/button&gt;
```

$ionicGesture

一个angular服务展示ionicionic.EventController手势。

方法

```
on(eventType, callback, $element)
```

在一个元素上添加一个事件监听器。

`eventType` : `string`

监听的手势事件。

`callback` : `function(e)`

当手势事件发生时触发的事件。

`$element` : `element`

angular元素监听的事件。

`options` : `object`

对象。

```
off(eventType, callback, $element)
```

在一个元素上移除一个手势事件监听器。

`eventType` : `string`

移除监听的手势事件。

`callback` : `function(e)`

移除监听器。

`$element` : `element`

被监听事件的angular元素。

ionic 头部和底部

ion-header-bar

这个是固定在屏幕顶部的一个头部标题栏。如果给它加上'bar-subheader' 这个样式，它就是副标题。

用法

```
<ion-header-bar align-title="left" class="bar-positive">
  <div class="buttons">
    <button class="button" ng-click="doSomething()">Left Button</bu
  </div>
  <h1 class="title">Title!</h1>
  <div class="buttons">
    <button class="button">Right Button</button>
  </div>
</ion-header-bar>
<ion-content>
  Some content!
</ion-content>
```

API

`align-title_(optional)_` : `string`

这个是对齐 title 的。如果没有设置，它将会按照手机的默认排版(Ios的默认是居中，Android默认是居左)。它的值可以是 'left','center','right'。

`no-tap-scroll_(optional)_` : `boolean`

这个是设置 header-bar 是否跟随着内容的滚动而滚动，就是是否固定在顶部。它的值是布尔值（true/false）。

ion-footer-bar

知道了 ion-header-bar， 理解ion-footer-bar就轻松多啦！只是 ion-footer-bar 是在屏幕的底部。

用法

```
<ion-content>
  Some content!
</ion-content>
<ion-footer-bar align-title="left" class="bar-assertive">
  <div class="buttons">
    <button class="button">Left Button</button>
  </div>
  <h1 class="title">Title!</h1>
  <div class="buttons" ng-click="doSomething()">
    <button class="button">Right Button</button>
  </div>
</ion-footer-bar>
```

API

与 ion-header-bar 不同的是，ion-footer-bar 只有 align-title 这个 API。

`align-title(optional)` : `string`

这个是对齐 title 的。如果没有设置，它将会按照手机的默认排版(Ios的默认是居中，Android默认是居左)。它的值可以是 'left','center','right'。

ionic 列表操作

列表是一个应用广泛在几乎所有移动app中的界面元素。ionList 和 ionItem 这两个指令还支持多种多样的交互模式，比如移除其中的某一项，拖动重新排序，滑动编辑等等。

用法

```
<ion-list>
  <ion-item ng-repeat="item in items">
    Hello, {{item}}!
  </ion-item>
</ion-list>
```

高级用法: 缩略图，删除按钮，重新排序，滑动

```
<ion-list ng-controller="MyCtrl"
          show-delete="shouldShowDelete"
          show-reorder="shouldShowReorder"
          can-swipe="listCanSwipe">
  <ion-item ng-repeat="item in items"
            class="item-thumbnail-left">

    <img ng-src="{{item.img}}">
    <h2>{{item.title}}</h2>
    <p>{{item.description}}</p>
    <ion-option-button class="button-positive"
                       ng-click="share(item)">
      分享
    </ion-option-button>
    <ion-option-button class="button-info"
                       ng-click="edit(item)">
      编辑
    </ion-option-button>
    <ion-delete-button class="ion-minus-circled"
                       ng-click="items.splice($index, 1)">
    </ion-delete-button>
    <ion-reorder-button class="ion-navicon"
                        on-reorder="reorderItem(item, $fromIndex, $
    </ion-reorder-button>

  </ion-item>
</ion-list>
```

API

`delegate-handle(可选)` : `字符串`

该句柄定义带有 `$ionicListDelegate` 的列表。

`show-delete(可选)` : `布尔值`

列表项的删除按钮当前是显示还是隐藏。

`show-reorder(可选)` : `布尔值`

列表项的排序按钮当前是显示还是隐藏。

`can-swipe(可选)` : `布尔值`

列表项是否被允许滑动显示选项按钮。默认：true。

实例

HTML 代码：

```
<html ng-app="ionicApp">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scal
    <title>Ionic List Directive</title>

    <link href="http://www.runoob.com/static/ionic/css/ionic.min.cs
    <script src="http://www.runoob.com/static/ionic/js/ionic.bundle
  </head>

  <body ng-controller="MyCtrl">

    <ion-header-bar class="bar-positive">
      <div class="buttons">
        <button class="button button-icon icon ion-ios-minus-outlin
          ng-click="data.showDelete = !data.showDelete; data.showRe
      </div>
      <h1 class="title">Ionic Delete/Option Buttons</h1>
      <div class="buttons">
        <button class="button" ng-click="data.showDelete = false; d
            Reorder
        </button>
      </div>
    </ion-header-bar>

    <ion-content>

      <!-- The list directive is great, but be sure to also checkou
```

```
      <ion-list show-delete="data.showDelete" show-reorder="data.sh

        <ion-item ng-repeat="item in items"
                  item="item"
                  href="#/item/{{item.id}}" class="item-remove-anir
          Item {{ item.id }}
          <ion-delete-button class="ion-minus-circled"
                             ng-click="onItemDelete(item)">
          </ion-delete-button>
          <ion-option-button class="button-assertive"
                             ng-click="edit(item)">
            Edit
          </ion-option-button>
          <ion-option-button class="button-calm"
                             ng-click="share(item)">
            Share
          </ion-option-button>
          <ion-reorder-button class="ion-navicon" on-reorder="moveI
        </ion-item>

      </ion-list>

    </ion-content>

  </body>
</html>
```

CSS 代码

```
body {
  cursor: url('http://www.runoob.com/try/demo_source/finger.png'),
}
```

JavaScript 代码

```
angular.module('ionicApp', ['ionic'])

.controller('MyCtrl', function($scope) {

  $scope.data = {
    showDelete: false
  };

  $scope.edit = function(item) {
    alert('Edit Item: ' + item.id);
```

```
  };
  $scope.share = function(item) {
    alert('Share Item: ' + item.id);
  };

  $scope.moveItem = function(item, fromIndex, toIndex) {
    $scope.items.splice(fromIndex, 1);
    $scope.items.splice(toIndex, 0, item);
  };

  $scope.onItemDelete = function(item) {
    $scope.items.splice($scope.items.indexOf(item), 1);
  };

  $scope.items = [
    { id: 0 },
    { id: 1 },
    { id: 2 },
    { id: 3 },
    { id: 4 },
    { id: 5 },
    { id: 6 },
    { id: 7 },
    { id: 8 },
    { id: 9 },
    { id: 10 },
    { id: 11 },
    { id: 12 },
    { id: 13 },
    { id: 14 },
    { id: 15 },
    { id: 16 },
    { id: 17 },
    { id: 18 },
    { id: 19 },
    { id: 20 },
    { id: 21 },
    { id: 22 },
    { id: 23 },
    { id: 24 },
    { id: 25 },
    { id: 26 },
    { id: 27 },
    { id: 28 },
    { id: 29 },
    { id: 30 },
    { id: 31 },
    { id: 32 },
    { id: 33 },
    { id: 34 },
    { id: 35 },
    { id: 36 },
    { id: 37 },
```

```
    { id: 38 },
    { id: 39 },
    { id: 40 },
    { id: 41 },
    { id: 42 },
    { id: 43 },
    { id: 44 },
    { id: 45 },
    { id: 46 },
    { id: 47 },
    { id: 48 },
    { id: 49 },
    { id: 50 }
  ];

});
```

ionic 加载动作

$ionicLoading 是 ionic 默认的一个加载交互效果。里面的内容也是可以在模板里面修改。

用法

```
angular.module('LoadingApp', ['ionic'])
.controller('LoadingCtrl', function($scope, $ionicLoading) {
  $scope.show = function() {
    $ionicLoading.show({
      template: 'Loading...'
    });
  };
  $scope.hide = function(){
    $ionicLoading.hide();
  };
});
```

方法

显示一个加载效果。

```
show(opts)
```

`opts` : `object`

loading指示器的选项。可用属性：

- `{string=}` `template` 指示器的html内容。
- `{string=}` `templateUrl` 一个加载html模板的url作为指示器的内容。
- `{boolean=}` `noBackdrop` 是否隐藏背景。默认情况下它会显示。
- `{number=}` `delay` 指示器延迟多少毫秒显示。默认为不延迟。
- `{number=}` `duration` 等待多少毫秒后自动隐藏指示器。默认情况下，指示器会一直显示，直到触发 `.hide()` 。

隐藏一个加载效果。

```
hide()
```

API

`delegate-handle(可选)` : `字符串`

该句柄定义带有 `$ionicListDelegate` 的列表。

`show-delete(可选)` : `布尔值`

列表项的删除按钮当前是显示还是隐藏。

`show-reorder(可选)` : `布尔值`

列表项的排序按钮当前是显示还是隐藏。

`can-swipe(可选)` : `布尔值`

列表项是否被允许滑动显示选项按钮。默认：true。

实例

HTML 代码：

```
<html ng-app="ionicApp">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="initial-scale=1, maximum-scale=1

    <title>Ionic Modal</title>

     <link href="http://www.runoob.com/static/ionic/css/ionic.min.c
    <script src="http://www.runoob.com/static/ionic/js/ionic.bundle
  </head>
  <body ng-controller="AppCtrl">

      <ion-view title="Home">
        <ion-header-bar>
          <h1 class="title">The Stooges</h1>
        </ion-header-bar>
        <ion-content has-header="true">
          <ion-list>
            <ion-item ng-repeat="stooge in stooges" href="#">{{stoo
          </ion-list>
        </ion-content>
      </ion-view>

  </body>
</html>
```

JavaScript 代码

```
angular.module('ionicApp', ['ionic'])
.controller('AppCtrl', function($scope, $timeout, $ionicLoading) {

  // Setup the loader
  $ionicLoading.show({
    content: 'Loading',
    animation: 'fade-in',
    showBackdrop: true,
    maxWidth: 200,
    showDelay: 0
  });

  // Set a timeout to clear loader, however you would actually call
  $timeout(function () {
    $ionicLoading.hide();
    $scope.stooges = [{name: 'Moe'}, {name: 'Larry'}, {name: 'Curly
  }, 2000);

});
```

$ionicLoadingConfig

设置加载的默认选项:

用法:

```
var app = angular.module('myApp', ['ionic'])
app.constant('$ionicLoadingConfig', {
  template: '默认加载模板……'
});
app.controller('AppCtrl', function($scope, $ionicLoading) {
  $scope.showLoading = function() {
    $ionicLoading.show(); //配置选项在 $ionicLoadingConfig 设置
  };
});
```

ionic 模型

$ionicModal

$ionicModal 可以遮住用户主界面的内容框。

你可以在你的 index 文件或者是其他文件内嵌入以下代码(里面的代码可以根据你自己的业务场景相应的改变)。

```
<script id="my-modal.html" type="text/ng-template">
  <ion-modal-view>
    <ion-header-bar>
      <h1 class="title">My Modal title</h1>
    </ion-header-bar>
    <ion-content>
      Hello!
    </ion-content>
  </ion-modal-view>
</script>
```

然后你就可以在你的 Controller 里面的注入 $ionicModal 。然后调用你刚刚写入的模板，进行初始化操作。就像下面的代码：

```
angular.module('testApp', ['ionic'])
.controller('MyController', function($scope, $ionicModal) {
  $ionicModal.fromTemplateUrl('my-modal.html', {
    scope: $scope,
    animation: 'slide-in-up'
  }).then(function(modal) {
    $scope.modal = modal;
  });
  $scope.openModal = function() {
    $scope.modal.show();
  };
  $scope.closeModal = function() {
    $scope.modal.hide();
  };
  //Cleanup the modal when we're done with it!
  $scope.$on('$destroy', function() {
    $scope.modal.remove();
  });
  // Execute action on hide modal
  $scope.$on('modal.hidden', function() {
    // Execute action
  });
  // Execute action on remove modal
  $scope.$on('modal.removed', function() {
    // Execute action
  });
});
```

方法

```
fromTemplate(templateString, options)
```

`templateString` : `字符串`

模板的字符串作为模型的内容。

`options` : `对象`

传递 ionicModal#initialize方法的选项。

返回: 对象, 一个ionicModal控制器的实例。

```
fromTemplateUrl(templateUrl, options)
```

`templateUrl` : `字符串`

载入模板的url。

`options` ： `对象`

通过ionicModal#initialize方法传递对象。

返回： promise对象。Promises对象是CommonJS工作组提出的一种规范，目的是为异步编程提供统一接口。

ionicModal

由$ionicModal服务实例化。

提示：当你完成每个模块清除时，确保调用remove()方法，以避免内存泄漏。

注意：一个模块从它的初始范围广播出 'modal.shown' 和 'modal.hidden'，把自身作为一个参数来传递。

方法

```
initialize(可选)
```

创建一个新的模型控制器示例。

`options` ： `对象`

一个选项对象具有一下属性：

- `{object=}` `范围` 子类的范围。默认：创建一个$rootScope子类。
- `{string=}` `动画` 带有显示或隐藏的动画。默认：'slide-in-up'
- `{boolean=}` `第一个输入框获取焦点` 当显示时，模型的第一个输入元素是否自动获取焦点。默认：false。
- `{boolean=}` backdropClickToClose` 点击背景时是否关闭模型。默认：true。

```
show()
```

显示模型实例

- 返回值: `promise` promise对象,在模型完成动画后得到解析

```
hide()
```

隐藏模型。

- 返回值: `promise` promise对象,在模型完成动画后得到解析

```
remove()
```

从 DOM 中移除模型实例并清理。

- 返回值: `promise` promise对象,在模型完成动画后得到解析

```
isShown()
```

- 返回：布尔值，用于判断模型是否显示。

实例

HTML 代码

```
<html ng-app="ionicApp">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="initial-scale=1, maximum-scale=1

    <title>菜鸟教程(runoob.com)</title>
    <link href="http://www.runoob.com/static/ionic/css/ionic.min.cs
    <script src="http://www.runoob.com/static/ionic/js/ionic.bundle
  </head>
  <body ng-controller="AppCtrl">

    <ion-header-bar class="bar-positive">
      <h1 class="title">Contacts</h1>
      <div class="buttons">
        <button class="button button-icon ion-compose" ng-click="mo
        </button>
      </div>
    </ion-header-bar>
    <ion-content>
      <ion-list>
        <ion-item ng-repeat="contact in contacts">
          {{contact.name}}
        </ion-item>
      </ion-list>
    </ion-content>

    <script id="templates/modal.html" type="text/ng-template">
      <ion-modal-view>
        <ion-header-bar class="bar bar-header bar-positive">
          <h1 class="title">New Contact</h1>
          <button class="button button-clear button-primary" ng-cli
        </ion-header-bar>
        <ion-content class="padding">
```

```
            <div class="list">
              <label class="item item-input">
                <span class="input-label">First Name</span>
                <input ng-model="newUser.firstName" type="text">
              </label>
              <label class="item item-input">
                <span class="input-label">Last Name</span>
                <input ng-model="newUser.lastName" type="text">
              </label>
              <label class="item item-input">
                <span class="input-label">Email</span>
                <input ng-model="newUser.email" type="text">
              </label>
              <button class="button button-full button-positive" ng-c
            </div>
          </ion-content>
        </ion-modal-view>
      </script>

  </body>
</html>
```

CSS 代码

```
body {
  cursor: url('http://www.runoob.com/try/demo_source/finger.png'),
}
```

JavaScript 代码

```
angular.module('ionicApp', ['ionic'])

.controller('AppCtrl', function($scope, $ionicModal) {

  $scope.contacts = [
    { name: 'Gordon Freeman' },
    { name: 'Barney Calhoun' },
    { name: 'Lamarr the Headcrab' },
  ];

  $ionicModal.fromTemplateUrl('templates/modal.html', {
    scope: $scope
  }).then(function(modal) {
    $scope.modal = modal;
  });

  $scope.createContact = function(u) {
    $scope.contacts.push({ name: u.firstName + ' ' + u.lastName });
    $scope.modal.hide();
  };

});
```

ionic 导航

ion-nav-view

当用户在你的app中浏览时，ionic能够检测到浏览历史。通过检测浏览历史，实现向左或向右滑动时可以正确转换视图。

采用AngularUI路由器模块等应用程序接口可以分为不同的$state(状态)。Angular的核心为路由服务，URLs可以用来控制视图。

AngularUI路由提供了一个更强大的状态管理，即状态可以被命名，嵌套， 以及合并视图，允许一个以上模板呈现在同一个页面。

此外，每个状态无需绑定到一个URL，并且数据可以更灵活地推送到每个状态。

以下实例中，我们将创建一个应用程序中包含不同状态的导航视图。

我们的标记中选择ionNavView作为顶层指令。显示一个页眉栏我们用 ionNavBar 指令通过导航更新。

接下来，我们需要设置我们的将渲染的状态值。

```
var app = angular.module('myApp', ['ionic']);
app.config(function($stateProvider) {
  $stateProvider
  .state('index', {
    url: '/',
    templateUrl: 'home.html'
  })
  .state('music', {
    url: '/music',
    templateUrl: 'music.html'
  });
});
```

再打开应用，$stateProvider 会查询url, 看是否匹配 index 状态值, 再加载index.html到<ion-nav-view>。

页面加载都是通过URLs配置的。在Angular中创建模板最一个简单的方式就是直接将他放到html模板文件中并且用

ionic 平台

$ionicPlatform

$ionicPlatform 用来检测当前的平台，以及诸如在PhoneGap/Cordova中覆盖Android后退按钮。

方法

```
onHardwareBackButton(callback)
```

有硬件的后退按钮的平台，可以用这种方法绑定到它。

`callback` : `function`

当该事件发生时，触发回调函数。

```
offHardwareBackButton(callback)
```

移除后退按钮的监听事件。

`callback` : `function`

最初绑定的监视器函数。

```
registerBackButtonAction(callback, priority, [actionId])
```

注册硬件后退按钮动作。当点击按钮时，只有一个动作会执行，因此该方法决定了注册的后退按钮动作具有最高的优先级。

例如，如果一个上拉菜单已经显示，后退按钮应该关闭上拉菜单，而不是返回一个页面视图或关闭一个打开的模型。

`callback` : `function`

当点击返回按钮时触发，如果该监视器具有最高的优先级。

`priority` : `number`

仅最高优先级的会执行。

`actionId(可选)` : `*`

该id指定这个动作。默认：一个随机且唯一的id。

返回值: 函数， 一个被触发的函数，将会注销 backButtonAction。

```
ready([callback])
```

设备准备就绪，则触发一个回调函数。

`callback(可选)` : `function=`

触发的函数。

返回: promise对象, 对象被构造 成功后得到解析。

ionic 浮动框

$ionicPopover

$ionicPopover 是一个可以浮在app内容上的一个视图框。

实例

HTML 代码

```
<p>
<button ng-click="openPopover($event)">打开浮动框</button>
</p>
<script id="my-popover.html" type="text/ng-template">
<ion-popover-view>
  <ion-header-bar>
    <h1 class="title">我的浮动框标题</h1>
  </ion-header-bar>
  <ion-content>
    Hello!
  </ion-content>
</ion-popover-view>
</script>
```

JavaScript 代码

```
angular.module('ionicApp', ['ionic'])
.controller( 'AppCtrl',['$scope','$ionicPopover','$timeout',functic

  $scope.popover = $ionicPopover.fromTemplateUrl('my-popover.html',
    scope: $scope
  });

  // .fromTemplateUrl() 方法
  $ionicPopover.fromTemplateUrl('my-popover.html', {
    scope: $scope
  }).then(function(popover) {
    $scope.popover = popover;
  });

  $scope.openPopover = function($event) {
    $scope.popover.show($event);
  };
  $scope.closePopover = function() {
    $scope.popover.hide();
  };
  // 清除浮动框
  $scope.$on('$destroy', function() {
    $scope.popover.remove();
  });
  // 在隐藏浮动框后执行
  $scope.$on('popover.hidden', function() {
    // 执行代码
  });
  // 移除浮动框后执行
  $scope.$on('popover.removed', function() {
    // 执行代码
  });

}])
```

ionic 对话框

$ionicPopup

ionic 对话框服务允许程序创建、显示弹出窗口。

$ionicPopup 提供了3个方法：alert(), prompt(),以及 confirm() 。

实例

HTML 代码

```
<body class="padding" ng-controller="PopupCtrl">
    <button class="button button-dark" ng-click="showPopup()">
      弹窗显示
    </button>
    <button class="button button-primary" ng-click="showConfirm()">
      确认对话框
    </button>
    <button class="button button-positive" ng-click="showAlert()">
      警告框
    </button>

    <script id="popup-template.html" type="text/ng-template">
      <input ng-model="data.wifi" type="text" placeholder="Passwor
    </script>
</body>
```

JavaScript 代码

```
angular.module('mySuperApp', ['ionic'])
.controller('PopupCtrl',function($scope, $ionicPopup, $timeout) {

 // Triggered on a button click, or some other target
 $scope.showPopup = function() {
   $scope.data = {}

   // 自定义弹窗
   var myPopup = $ionicPopup.show({
     template: '<input type="password" ng-model="data.wifi">',
     title: 'Enter Wi-Fi Password',
     subTitle: 'Please use normal things',
     scope: $scope,
```

```
      buttons: [
        { text: 'Cancel' },
        {
          text: '<b>Save</b>',
          type: 'button-positive',
          onTap: function(e) {
            if (!$scope.data.wifi) {
              // 不允许用户关闭，除非输入 wifi 密码
              e.preventDefault();
            } else {
              return $scope.data.wifi;
            }
          }
        },
      ]
    });
    myPopup.then(function(res) {
      console.log('Tapped!', res);
    });
    $timeout(function() {
       myPopup.close(); // 3秒后关闭弹窗
    }, 3000);
   };
    //  confirm 对话框
    $scope.showConfirm = function() {
      var confirmPopup = $ionicPopup.confirm({
        title: 'Consume Ice Cream',
        template: 'Are you sure you want to eat this ice cream?'
      });
      confirmPopup.then(function(res) {
        if(res) {
          console.log('You are sure');
        } else {
          console.log('You are not sure');
        }
      });
    };

    //  alert（警告） 对话框
    $scope.showAlert = function() {
      var alertPopup = $ionicPopup.alert({
        title: 'Don\'t eat that!',
        template: 'It might taste good'
      });
      alertPopup.then(function(res) {
        console.log('Thank you for not eating my delicious ice crear
      });
    };
 });
```

ionic 滚动条

ion-scroll

ion-scroll 用于创建一个可滚动的容器。

用法

```
<ion-scroll
    [delegate-handle=""]
    [direction=""]
    [paging=""]
    [on-refresh=""]
    [on-scroll=""]
    [scrollbar-x=""]
    [scrollbar-y=""]
    [zooming=""]
    [min-zoom=""]
    [max-zoom=""]>
    ...
</ion-scroll>
```

API

`delegate-handle(可选)` : `字符串`

该句柄利用 `$ionicScrollDelegate` 指定滚动视图。

`direction(可选)` : `字符串`

滚动的方向。 'x' 或 'y'。 默认 'y'。

`paging(可选)` : `布尔值`

分页是否滚动。

`on-refresh(可选)` : `表达式`

调用下拉刷新， 由 `ionRefresher` 触发。

`on-scroll(可选)` : `表达式`

当用户滚动时触发。

`scrollbar-x(可选)` : `布尔值`

是否显示水平滚动条。默认为false。

`scrollbar-y(可选)` : `布尔值`

是否显示垂直滚动条。默认为true。

`zooming(可选)` : `布尔值`

是否支持双指缩放。

`min-zoom(可选)` : `整数`

允许的最小缩放量（默认为0.5）

`max-zoom(可选)` : `整数`

允许的最大缩放量（默认为3）

实例

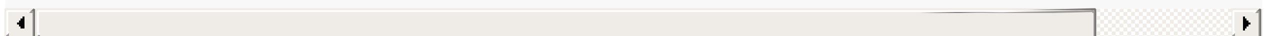
HTML 代码

```
<ion-scroll zooming="true" direction="xy" style="width: 500px; heig
    <div style="width: 5000px; height: 5000px; background: url('htt
</ion-scroll>
```



CSS 代码

```
body {
  cursor: url('http://www.runoob.com/try/demo_source/finger.png'),
}
```



JavaScript 代码

```
angular.module('ionicApp', ['ionic']);
```

ion-infinite-scroll

当用户到达页脚或页脚附近时，ionInfiniteScroll指令允许你调用一个函数 。

当用户滚动的距离超出底部的内容时，就会触发你指定的on-infinite。

用法

```
<ion-content ng-controller="MyController">
  <ion-infinite-scroll
    on-infinite="loadMore()"
    distance="1%">
  </ion-infinite-scroll>
</ion-content>
```

```
function MyController($scope, $http) {
  $scope.items = [];
  $scope.loadMore = function() {
    $http.get('/more-items').success(function(items) {
      useItems(items);
      $scope.$broadcast('scroll.infiniteScrollComplete');
    });
  };

  $scope.$on('stateChangeSuccess', function() {
    $scope.loadMore();
  });
}
```

当没有更多数据加载时，就可以用一个简单的方法阻止无限滚动，那就是angular的ng-if 指令:

```
<ion-infinite-scroll
  ng-if="moreDataCanBeLoaded()"
  icon="ion-loading-c"
  on-infinite="loadMoreData()">
</ion-infinite-scroll>
```

API

`on-infinite` ： `表达式`

当滚动到底部时触发的时间。

`distance(可选)` ： `字符串`

从底部滚动到触发on-infinite表达式的距离。默认: 1%。

`icon(可选)` ： `字符串`

当加载时显示的图标。默认: 'ion-loading-d'。

$ionicScrollDelegate

授权控制滚动视图（通过ion-content 和 ion-scroll指令创建）。

该方法直接被$ionicScrollDelegate服务触发，来控制所有滚动视图。用$getByHandle方法控制特定的滚动视图。

用法

```
<body ng-controller="MainCtrl">
  <ion-content>
    <button ng-click="scrollTop()">滚动到顶部!</button>
  </ion-content>
</body>
```

```
function MainCtrl($scope, $ionicScrollDelegate) {
  $scope.scrollTop = function() {
    $ionicScrollDelegate.scrollTop();
  };
}
```

方法

```
resize()
```

告诉滚动视图重新计算它的容器大小。

```
scrollTop([shouldAnimate])
```

`shouldAnimate(可选)` ： `布尔值`

是否应用滚动动画。

```
scrollBottom([shouldAnimate])
```

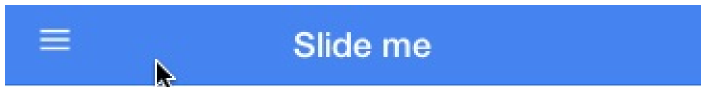
`shouldAnimate(可选)` ： `布尔值`

是否应用滚动动画。

ionic 侧栏菜单

一个容器元素包含侧边菜单和主要内容。通过把主要内容区域从一边拖动到另一边，来让左侧或右侧的侧栏菜单进行切换。

效果图如下所示：



Content

用法

要使用侧栏菜单，添加一个父元素<ion-side-menus>，一个中间内容 <ion-side-menu-content>，和一个或更多 <ion-side-menu> 指令。

```
<ion-side-menus>
  <!-- 中间内容 -->
  <ion-side-menu-content ng-controller="ContentController">
  </ion-side-menu-content>

  <!-- 左侧菜单 -->
  <ion-side-menu side="left">
  </ion-side-menu>

  <!-- 右侧菜单 -->
  <ion-side-menu side="right">
  </ion-side-menu>
</ion-side-menus>
```

```
function ContentController($scope, $ionicSideMenuDelegate) {
  $scope.toggleLeft = function() {
    $ionicSideMenuDelegate.toggleLeft();
  };
}
```

API

`enable-menu-with-back-views(可选)` : `布尔值`

在返回按钮显示时，确认是否启用侧边栏菜单。

`delegate-handle` : 字符串 该句柄用于标识带有$ionicScrollDelegate的滚动视图。

ion-side-menu-content

一个可见主体内容的容器，同级的一个或多个ionSideMenu 指令。

用法

```
<ion-side-menu-content
  drag-content="true">
</ion-side-menu-content>
```

API

`drag-content(可选)` : `布尔值`

内容是否可被拖动。默认为true。

ion-side-menu

一个侧栏菜单的容器，同级的一个ion-side-menu-content 指令。

用法

```
<ion-side-menu
  side="left"
  width="myWidthValue + 20"
  is-enabled="shouldLeftSideMenuBeEnabled()">
</ion-side-menu>
```

API

`side` : `字符串`

侧栏菜单当前在哪一边。可选的值有: 'left' 或 'right'。

`is-enabled(可选)` : `布尔值`

该侧栏菜单是否可用。

`width(可选)` : `数值`

侧栏菜单应该有多少像素的宽度。默认为275。

menu-toggle

在一个指定的侧栏中切换菜单。

用法

下面是一个在导航栏内链接的例子。点击此链接会自动打开指定的侧栏菜单。

```
<ion-view>
  <ion-nav-buttons side="left">
   <button menu-toggle="left" class="button button-icon icon ion-na
  </ion-nav-buttons>
 ...
</ion-view>
```

menu-close

关闭当前打开的侧栏菜单。

用法

下面是一个在导航栏内链接的例子。点击此链接会自动打开指定的侧栏菜单。

```
<a menu-close href="#/home" class="item">首页</a>
```

$ionicSideMenuDelegate

该方法直接触发$ionicSideMenuDelegate服务，来控制所有侧栏菜单。用$getByHandle方法控制特定情况下的ionSideMenus。

用法

```
<body ng-controller="MainCtrl">
  <ion-side-menus>
    <ion-side-menu-content>
      内容!
      <button ng-click="toggleLeftSideMenu()">
        切换左侧侧栏菜单
      </button>
    </ion-side-menu-content>
    <ion-side-menu side="left">
      左侧菜单!
    <ion-side-menu>
  </ion-side-menus>
</body>
```

```
function MainCtrl($scope, $ionicSideMenuDelegate) {
  $scope.toggleLeftSideMenu = function() {
    $ionicSideMenuDelegate.toggleLeft();
  };
}
```

方法

```
toggleLeft([isOpen])
```

切换左侧侧栏菜单（如果存在）。

`isOpen(可选)` : `布尔值`

是否打开或关闭菜单。默认：切换菜单。

```
toggleRight([isOpen])
```

切换右侧侧栏菜单（如果存在）。

`isOpen(可选)` ： `布尔值`

是否打开或关闭菜单。默认：切换菜单。

```
getOpenRatio()
```

获取打开菜单内容超出菜单宽度的比例。比如，一个宽度为100px的菜单被宽度为50px以50%的比例打开，将会返回一个比例值为0.5。

返回值: 浮点 0 表示没被打开，如果左侧菜单处于已打开或正在打开为0 到 1，如果右侧菜单处于已打开或正在打开为0 到-1。

```
isOpen()
```

返回值: 布尔值，判断左侧或右侧菜单是否已经打开。

```
isOpenLeft()
```

返回值: 布尔值左侧菜单是否已经打开。

```
isOpenRight()
```

返回值: 布尔值右侧菜单是否已经打开。

```
canDragContent([canDrag])
```

`canDrag(可选)` ： `布尔值`

设置是否可以拖动内容打开侧栏菜单。

返回值: 布尔值，是否可以拖动内容打开侧栏菜单。

```
$getByHandle(handle)
```

`handle` ： `字符串`

例如:

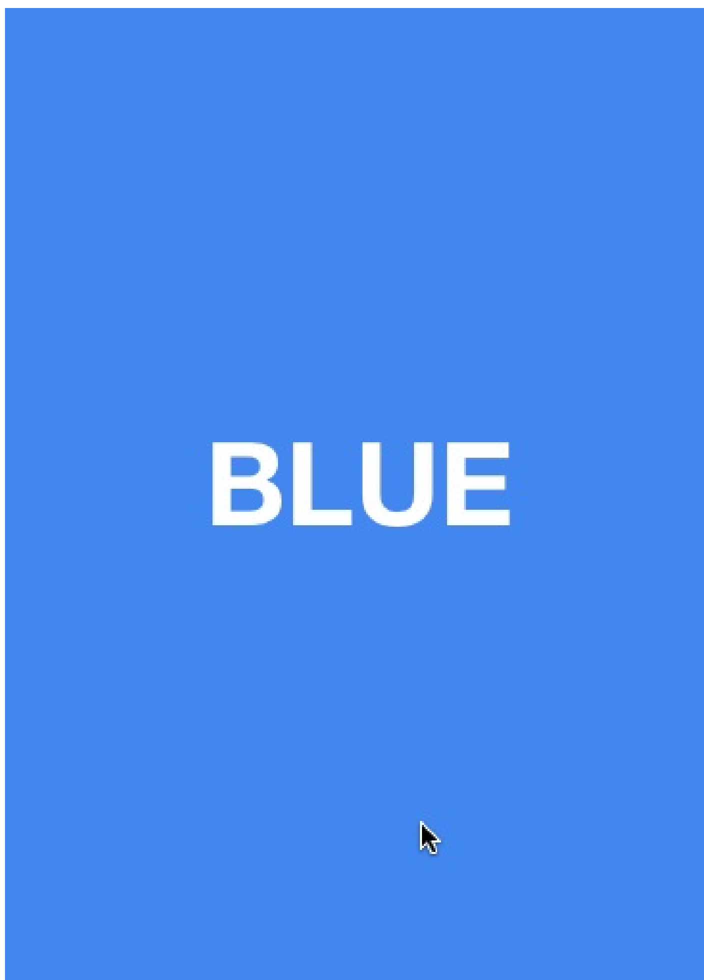
```
$ionicSideMenuDelegate.$getByHandle('my-handle').toggleLeft();
```

ionic 滑动框

ion-slide-box

滑动框是一个包含多页容器的组件，每页滑动或拖动切换：

效果图如下：



用法

```
<ion-slide-box on-slide-changed="slideHasChanged($index)">
  <ion-slide>
    <div class="box blue"><h1>BLUE</h1></div>
  </ion-slide>
  <ion-slide>
    <div class="box yellow"><h1>YELLOW</h1></div>
  </ion-slide>
  <ion-slide>
    <div class="box pink"><h1>PINK</h1></div>
  </ion-slide>
</ion-slide-box>
```

API

`delegate-handle(可选)` : `字符串`

该句柄用 `$ionicSlideBoxDelegate` 来标识这个滑动框。

`does-continue(可选)` : `布尔值`

滑动框是否自动滑动。

`slide-interval(可选)` : `数字`

等待多少毫秒开始滑动（如果继续则为true）。默认为4000。

`show-pager(可选)` : `布尔值`

滑动框的页面是否显示。

`pager-click(可选)` : `表达式`

当点击页面时，触发该表达式（如果shou-pager为true）。传递一个'索引'变量。

`on-slide-changed(可选)` : `表达式`

当滑动时，触发该表达式。传递一个'索引'变量。

`active-slide(可选)` : `表达式`

将模型绑定到当前滑动框。

实例

HTML 代码

```
<ion-slide-box active-slide="myActiveSlide">
    <ion-slide>
      <div class="box blue"><h1>BLUE</h1></div>
    </ion-slide>
    <ion-slide>
      <div class="box yellow"><h1>YELLOW</h1></div>
    </ion-slide>
    <ion-slide>
      <div class="box pink"><h1>PINK</h1></div>
    </ion-slide>
</ion-slide-box>
```

CSS 代码

```
.slider {
  height: 100%;
}
.slider-slide {
  padding-top: 80px;
  color: #000;
  background-color: #fff;
  text-align: center;

  font-family: "HelveticaNeue-Light", "Helvetica Neue Light", "Helv
  font-weight: 300;
}

.blue {
  background-color: blue;
}

.yellow {
  background-color: yellow;
}

.pink {
  background-color: pink;
}
```

JavaScript 代码

```
angular.module('ionicApp', ['ionic'])

.controller('SlideController', function($scope) {

  $scope.myActiveSlide = 1;

})
```

ionic 加载动画

ion-spinner

ionSpinner 提供了许多种旋转加载的动画图标。当你的界面加载时，你就可以呈现给用户相应的加载图标。

该图标采用的是SVG。

用法

```
<ion-spinner icon="spiral"></ion-spinner>    //默认用法
```

像大部分其他的ionic组件一样，spinner也可以使用ionic的标准颜色命名规则，就像下面这样：

```
<ion-spinner class="spinner-energized"></ion-spinner>
```

实例

HTML 代码

```
<ion-content scroll="false" class="has-header">
  <p>
    <ion-spinner icon="android"></ion-spinner>
    <ion-spinner icon="ios"></ion-spinner>
    <ion-spinner icon="ios-small"></ion-spinner>
    <ion-spinner icon="bubbles" class="spinner-balanced"></ion-spir
    <ion-spinner icon="circles" class="spinner-energized"></ion-sp
  </p>

  <p>
    <ion-spinner icon="crescent" class="spinner-royal"></ion-spinne

    <ion-spinner icon="dots" class="spinner-dark"></ion-spinner>
    <ion-spinner icon="lines" class="spinner-calm"></ion-spinner>
    <ion-spinner icon="ripple" class="spinner-assertive"></ion-spir
    <ion-spinner icon="spiral"></ion-spinner>
  </p>

</ion-content>
```

CSS 代码

```
body {
  cursor: url('http://www.runob.com/try/demo_source/finger.png'), a
}
p {
  text-align: center;
  margin-bottom: 40px !important;
}
.spinner svg {
  width: 19% !important;
  height: 85px !important;
}
```

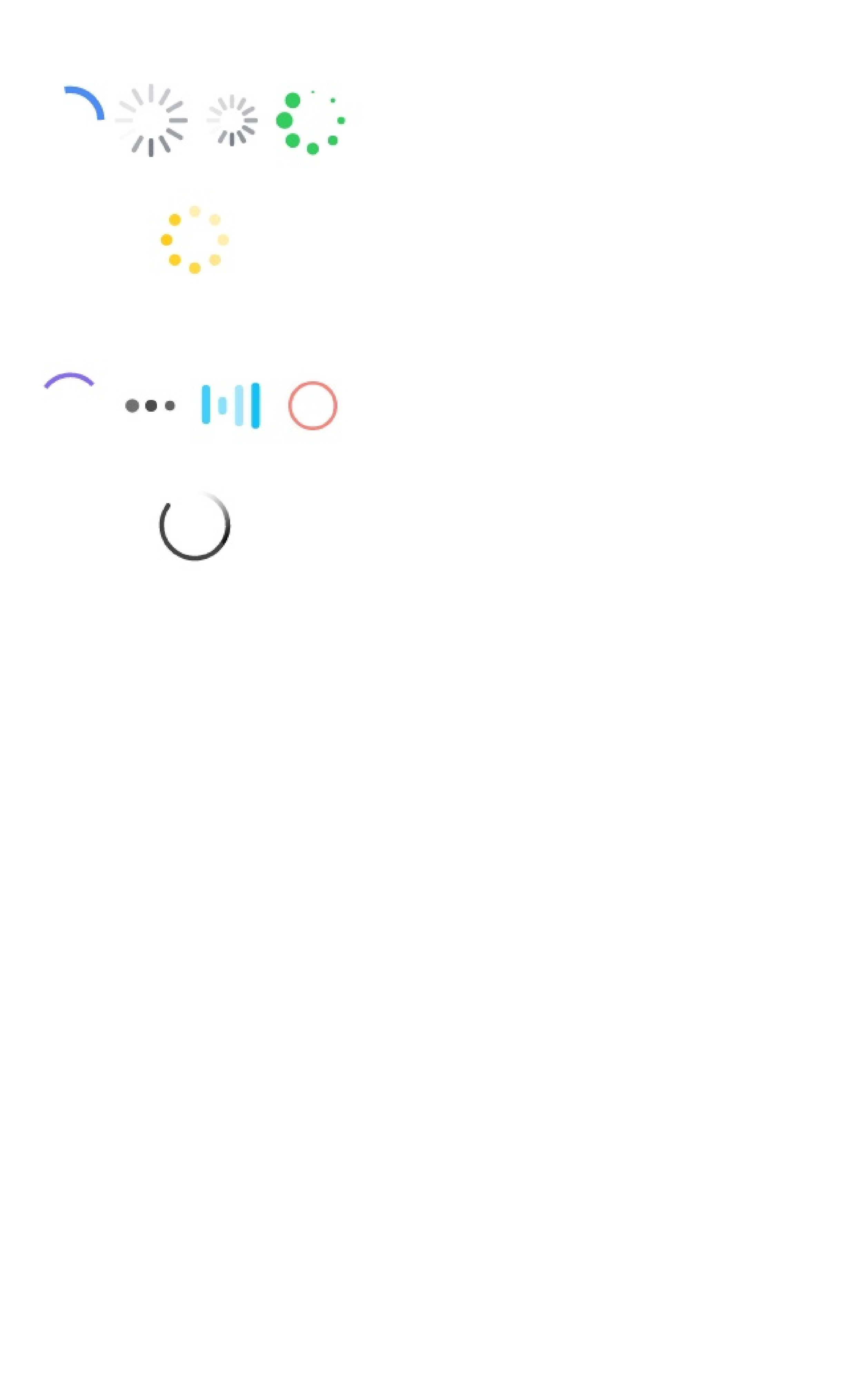
JavaScript 代码

```
angular.module('ionicApp', ['ionic'])

.controller('MyCtrl', function($scope) {

});
```

效果如下所示：



ionic 选项卡栏操作

ion-tabs

ion-tabs 是有一组页面选项卡组成的选项卡栏。可以通过点击选项来切换页面。

对于 iOS，它会出现在屏幕的底部，Android会出现在屏幕的顶部(导航栏下面)。

用法

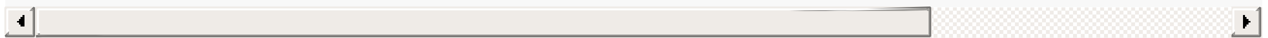
```
<ion-tabs class="tabs-positive tabs-icon-only">

  <ion-tab title="首页" icon-on="ion-ios7-filing" icon-off="ion-ios
    <!-- 标签 1 内容 -->
  </ion-tab>

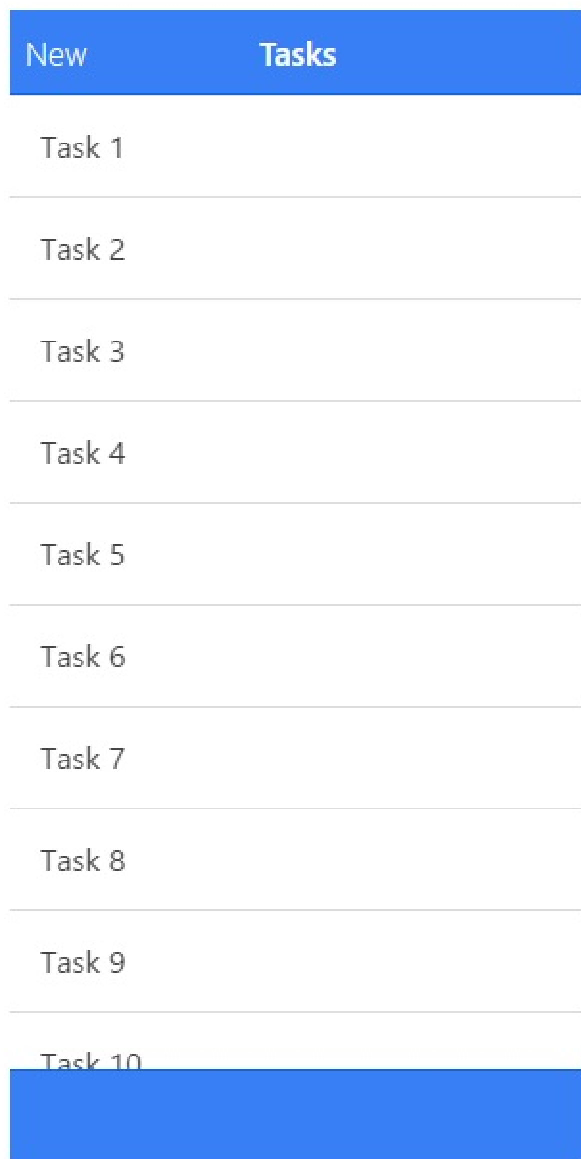
  <ion-tab title="关于" icon-on="ion-ios7-clock" icon-off="ion-ios7
    <!-- 标签 2 内容 -->
  </ion-tab>

  <ion-tab title="设置" icon-on="ion-ios7-gear" icon-off="ion-ios7-
    <!-- 标签 3 内容 -->
  </ion-tab>

</ion-tabs>
```



效果如下所示：



API

`delegate-handle(可选)` : `字符串`

该句柄用 `$ionicTabsDelegate` 来标识这些选项卡。

ion-tab

隶属于ionTabs

包含一个选项卡内容。该内容仅存在于被选中的给定选项卡中。

每个ionTab都有自己的浏览历史。

用法

```
<ion-tab
  title="Tab!"
  icon="my-icon"
  href="#/tab/tab-link"
  on-select="onTabSelected()"
  on-deselect="onTabDeselected()">
</ion-tab>
```

API

`title` ： `字符串`

选项卡的标题。

`href(可选)` ： `字符串`

但触碰的时候，该选项卡将会跳转的的链接。

`icon(可选)` ： `字符串`

选项卡的图标。如果给定值，它将成为ion-on和ion-off的默认值。

`icon-on(可选)` ： `字符串`

被选中标签的图标。

`icon-off(可选)` ： `字符串`

没被选中标签的图标。

`badge(可选)` ： `表达式`

选项卡上的徽章（通常是一个数字）。

`badge-style(可选)` ： `表达式`

选项卡上微章的样式（例，tabs-positive ）。

`on-select(可选)` ： `表达式`

选项卡被选中时触发。

`on-deselect(可选)` ： `表达式`

选项卡取消选中时触发。

`ng-click(可选)` ： `表达式`

通常，点击时选项卡会被选中。如果设置了 ng-Click，它将不会被选中。 你可以用$ionicTabsDelegate.select()来指定切换标签。

$ionicTabsDelegate

授权控制ionTabs指令。

该方法直接调用$ionicTabsDelegate服务，控制所有ionTabs指令。用$getByHandle方法控制具体的ionTabs实例。

用法

```
<body ng-controller="MyCtrl">
  <ion-tabs>

    <ion-tab title="Tab 1">
      你好，标签1！
      <button ng-click="selectTabWithIndex(1)">选择标签2</button>
    </ion-tab>
    <ion-tab title="Tab 2">你好标签2！</ion-tab>

  </ion-tabs>
</body>
```

```
function MyCtrl($scope, $ionicTabsDelegate) {
  $scope.selectTabWithIndex = function(index) {
    $ionicTabsDelegate.select(index);
  }
}
```

方法

```
select(index, [shouldChangeHistory])
```

选择标签来匹配给定的索引。

`index` ： `数值`

选择标签的索引。

`shouldChangeHistory(可选)` ： `布尔值`

此选项是否应该加载这个标签的浏览历史（如果存在），并使用，或仅加载默认页面。默认为false。提示：如果一个 `ion-nav-view` 在选项卡里，你可能需要设置它为true。

```
selectedIndex()
```

返回值: 数值, 被选中标签的索引，如 -1。

```
$getByHandle(handle)
```

`handle` : `字符串`

例如:

```
$ionicTabsDelegate.$getByHandle('my-handle').select(0);
```

W3School ios教程

来源：ios教程

整理：飞龙

IOS 简介

IOS之前被称为iPhone OS，是一个由苹果公司开发的移动操作系统。

iOS的第一个版本是在2007年发布的，其中包括iPhone和iPod Touch。

2004年4月发布iPad（第一代），并于2012年11月发布了iPad迷你款。

IOS设备发布相当频繁，由以往经验可知，每年都会推出至少一个版本的iPhone和iPad。

现在发布了iPhone5s，之前还推出了iPhone，iPhone3gs，iPhone4,iPhone4s以及iphone5。

同样的iPad也从iPad一代更新到iPad四代以及一个特别的迷你版iPad。

iOS SDK已经从1.0更新到6.0。最新的iOS SDK6.0，是唯一支持Xcode4.5和其更高版本的版本。

丰富的苹果文档，使我们能找到许多方法和库用于我们的部署目标。在Xcode的当前版本中，我们能够在iOS4.3,5.0和6.0的部署目标之间选择。

IOS的影响能够从以下的特点显现:

Facebook和Twitter上，加速度计，GPS，高端处理器，相机，Safari浏览器，功能强大的API，游戏中心，在应用程序内购买，提醒，宽范围的手势

- 地图
- Siri
- Facebook 和 Twitter
- Multi-Touch（多点触摸）
- Accelerometer（加速度传感器）
- GPS
- 高性能处理器
- 相机
- Safari浏览器
- 功能强大的API
- 游戏中心
- 在应用程序内购买
- 提醒功能
- 手势

iPhone和iPad的用户日益增多，这为iPhone和iPad应用商城的研发者创造了赚钱的机遇。

IOS最新的一点是，苹果公司研发了应用商城，这样用户可以购买应用程序来完善他们的iOS设备。

研发者可以在应用商城发布免费和付费的应用软件。

开发应用程序并将其发布到应用商店，开发人员需要注册iOS开发者计划，为其发展更新Xcode每年话费99美元和Mac Mountain Lion 或更高。

注册Apple开发者

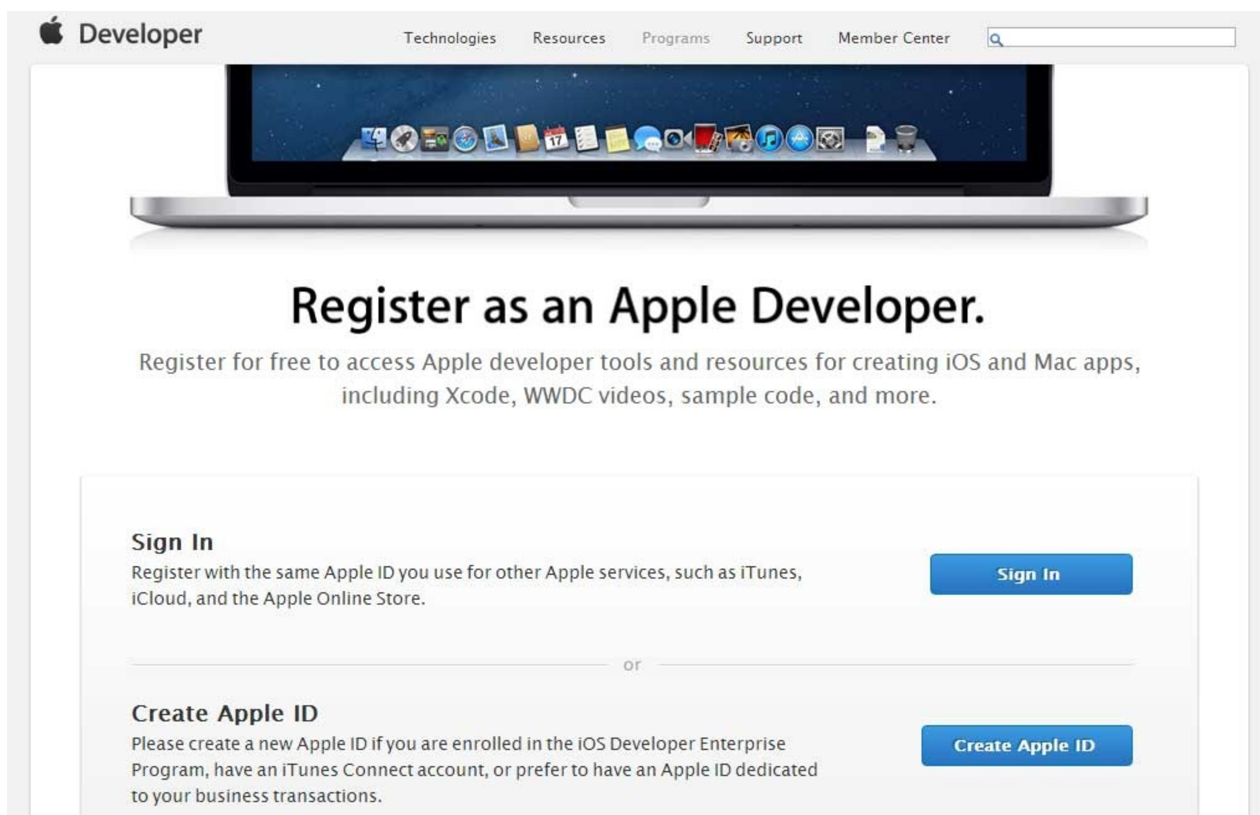
对拥有Apple设备的用户来说，非常有必要拥有Apple ID，而且成为一个研发者，必须用到Apple ID,获取 Apple ID是免费的，也无需有资费方面的顾虑。

拥有Apple账户有以下好处：

- 易于了解研发工具；
- 全球研发者视频会议；
- 受邀加入iOS研发者团队;

注册苹果账号

1、单击 (https://developer.apple.com/programs/register/) 并选择创建Apple ID



2、输入个人信息

3、返回邮箱确认，激活账号

4、下载研发工具，Xcode及它所包含的iOS模拟器，iOS SDK和其他研发资源

申请APP开发者

1、点击 (https://developer.apple.com/programs/ios/)



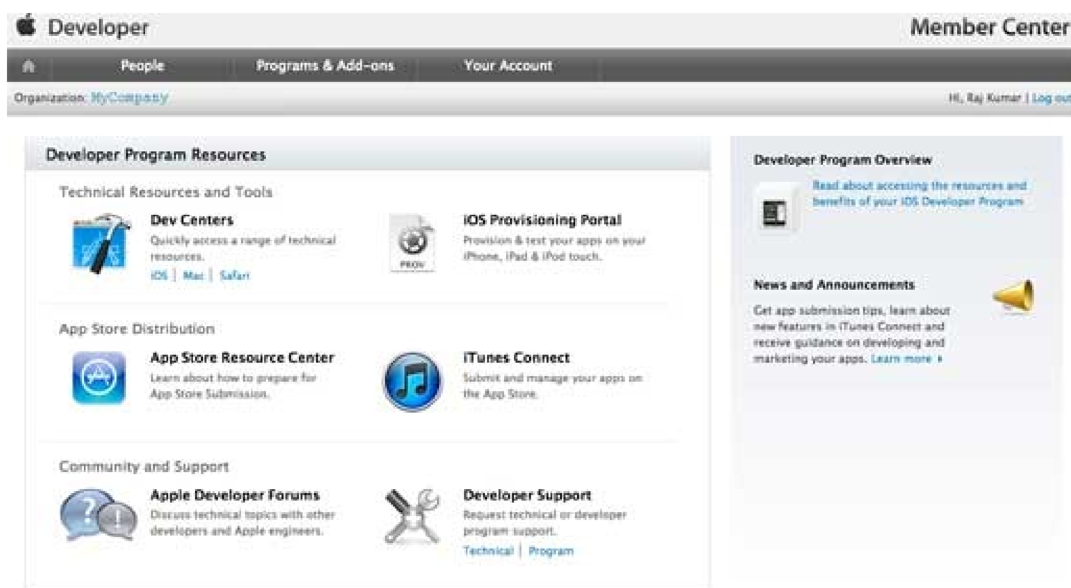
2、点击注册页面

3、登录账号（已有账号）或注册Apple ID

4、选择个人账号或公司账号，研发者团队使用公司账号，个人账号不能添加其他用户

5、新用户进入个人信息页面，使用信用卡购买加入研发项目

6、选择会员中心，利用研发者资源



7、在此处可以执行以下操作:

- 创建资源调配的配置文件
- 管理团队和设备
- 通过iTunes Connect管理应用到应用程序
- 获取论坛和技术支持

IOS Xcode 安装

1、从 https://developer.apple.com/downloads/ 下载Xcode的最新版本。



2、双击Xcode dmg文件

3、将找到的设备安装和打开

4、在这里会有两个项目在显示的窗口中即Xcode应用程序和应用程序文件夹的快捷方式

5、将Xcode拖拽并复制到应用程序

6、在应用里选择和运行程序，Xcode也将成为运行程序中的一部分

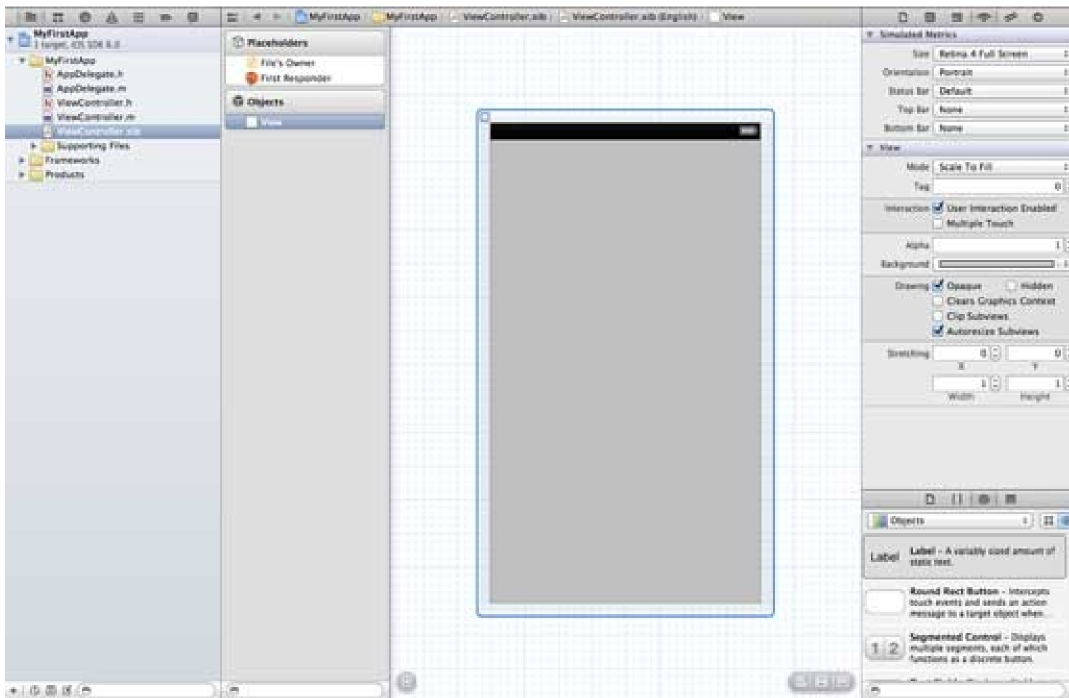
还可以从Mac App store里下载Xcode，并按照屏幕上的安装步

界面生成器（Interface Builder）

利用界面生成器这一工具，能很容易的创建UI界面。

可利用一系列的UI元素，拖拽进入UI可视界面。

我们将在接下来的页面了解添加用户界面元素，创建零售商和UI元素的操作。



在对象库的下方包含有全部必要的UI元素。用户界面通常称为xibs，这是他们的文件扩展名。

每个xibs都链接到相应的试图控制器。

IOS模拟器

IOS模拟器实际上包含两种类型的设备即iPhone和iPad及其不同的版本。

iPhone版本包括iPhone（常规版）、iPhone Retina，iPhone5,iPhone53。

Ipad有iPad和iPad Retina。iPhone模拟器显示如下:



你可以在经度和纬度影响应用程序的位置的情况下运行iOS模拟器，也可以模拟内存警告和呼叫在模拟器中的状态。

能够多数目的使用模拟器，但不能测试像加速度计这样的设备的功能。因此你可能需要iOS设备来测试一个应用程序的所有方面。

Objective-C 简介

在iOS的开发中使用的是Objective C语言，它是一中面向对象的语言，因而对于已经掌握面向对象语言知识的编程者来说是非常简单的。

接口和实现

在Objective里完成的文件被称为界面文件，该类文件的定义被称为实现文件。

一个简单的界面文件MyClass.h将如图所示:

```
@interace MyClass:NSObject{
// class variable declared here
}
// class properties declared here
// class methods and instance methods declared here
@end
```

执行MyClass.m文件，如下所示

```
@implementation MyClass
// class methods defined here
@end
```

创建对象

完成创建对象，如下所示

```
MyClass  *objectName = [[MyClass alloc]init] ;
```

方法（methods）

Objective C中声明的方法如下所示:

```
-(returnType)methodName:(typeName) variable1 :(typeName)variable2;
```

下面显示了一个示例:

```
-(void)calculateAreaForRectangleWithLength:(CGfloat)length
andBreadth:(CGfloat)breadth;
```

你可能会想什么是andBreadth字符串，其实它的可选字符串可以帮助我们阅读和理解方法，尤其是当方法被调用的时候。

在同一类中调用此方法，我们使用下面的语句。

```
[self calculateAreaForRectangleWithLength:30 andBreadth:20];
```

正如上文所说的andBreath使用有助于我们理解breath是20。Self用来指定它是一个类的方法。

类方法（**class methods**）

直接而无需创建的对象，可以访问类方法。他们没有任何变量和它关联的对象。示例如下:

```
+(void)simpleClassMethod;
```

它可以通过使用类名（假设作为MyClass类名称）访问，如下所示:

```
[MyClass simpleClassMethod];
```

实例方法

可以创建的类的对象后只访问实例方法，内存分配到的实例变量。实例方法如下所示:

```
-(void)simpleInstanceMethod;
```

创建类的对象后，它可以访问它。如下所示:

```
MyClass  *objectName = [[MyClass alloc]init] ;
[objectName simpleInstanceMethod];
```

Objective C的重要数据类型

数据类型
NSString 字符串
CGfloat 浮点值的基本类型
NSInteger 整型
BOOL 布尔型

打印日志

NSLog用于打印一份声明，它将打印在设备日志和调试版本的控制台和分别调试模式上。

如 NSlog(@"");

控制结构

除了几个增补的条款外，大多数的控制结构与C以及C++相同

属性（**properties**）

用于访问类的外部类的变量属性

比如：@property（非原子、强）NSString*myString

访问属性

可以使用点运算符访问属性，若要访问上一属性可以执行以下操作

```
self.myString = @"Test";
```

还可以使用set的方法，如下所示：

```
[self setMyString:@"Test"];
```

类别（**categories**）

类用于将方法添加到现有类。通过这种方法可以将方法添加到类，甚至不用执行文件，就可以在其中定义实际的类。MyClass的样本类别，如下所示：

```
@interace MyClass(customAdditions)
- (void)sampleCategoryMethod;
@end

@implementation MyClass(categoryAdditions)

-(void)sampleCategoryMethod{
   NSLog(@"Just a test category");
}
```

数组（arrays）

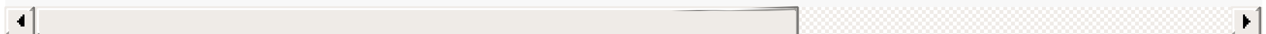
NSMutable和NSArray 是ObjectiveC中使用的数组类，前者是可变数组，后者是不可变数组。如下:

```
NSMutableArray *aMutableArray = [[NSMutableArray alloc]init];
[anArray addObject:@"firstobject"];
NSArray *aImmutableArray = [[NSArray alloc]
initWithObjects:@"firstObject",nil];
```

词典

NSMutableDictionary和NSDictionary是Objective中使用的字典，前者可变词典，后者不可变词典，如下所示:

```
NSMutableDictionary*aMutableDictionary = [[NSMutableArray alloc]ini
[aMutableDictionary setObject:@"firstobject" forKey:@"aKey"];
NSDictionary*aImmutableDictionary= [[NSDictionary alloc]initWithObj
@"firstObject",nil] forKeys:[ NSArray arrayWithObjects:@"aKey"]];
```



创建第一款iPhone应用程序

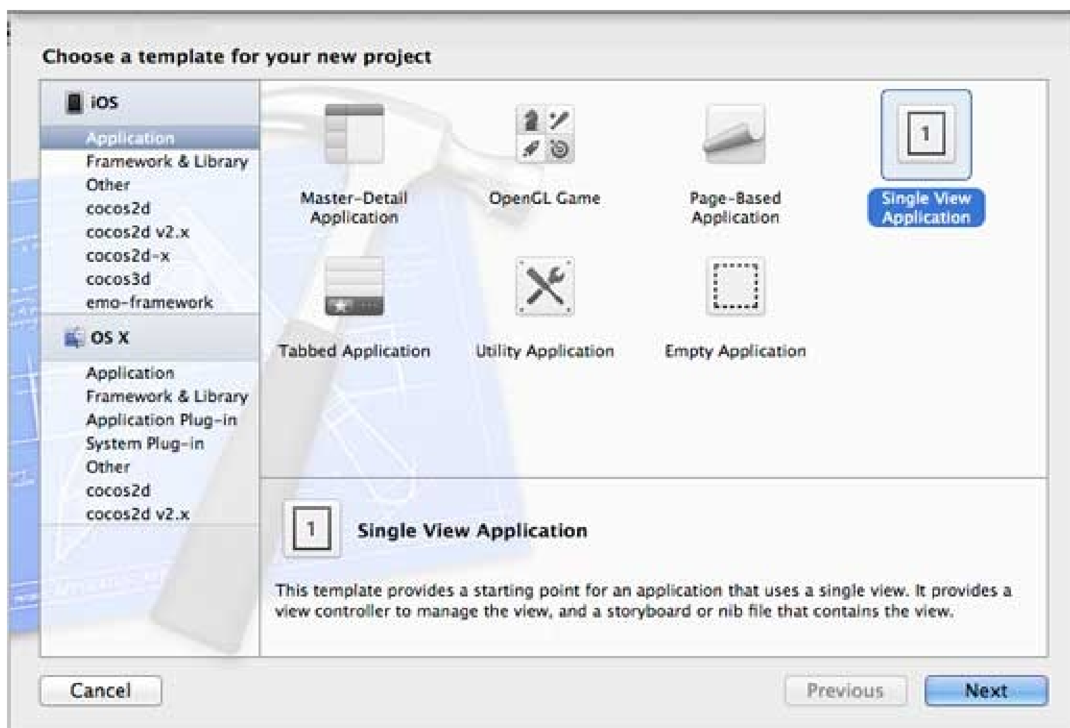
现在让我们来创建一个在iOS模拟器上运行的简单视图应用（空白的应用程序）。

操作步骤如下：

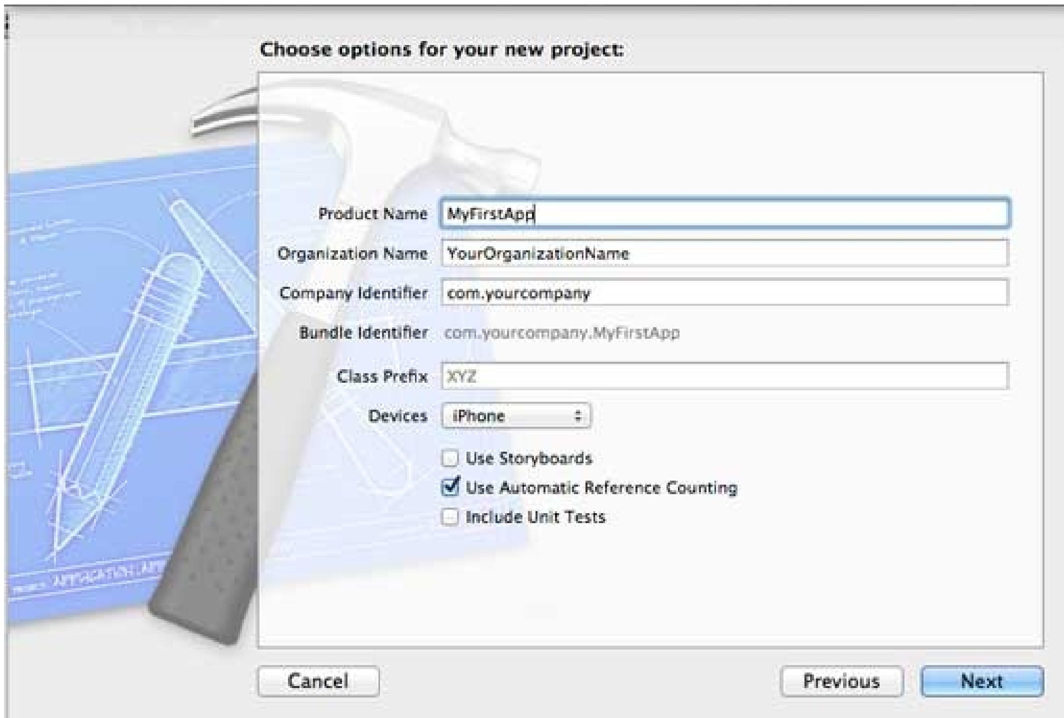
1、打开Xcode并选择创建一个新的Xcode项目。



2. 然后选择单一视图应用程序

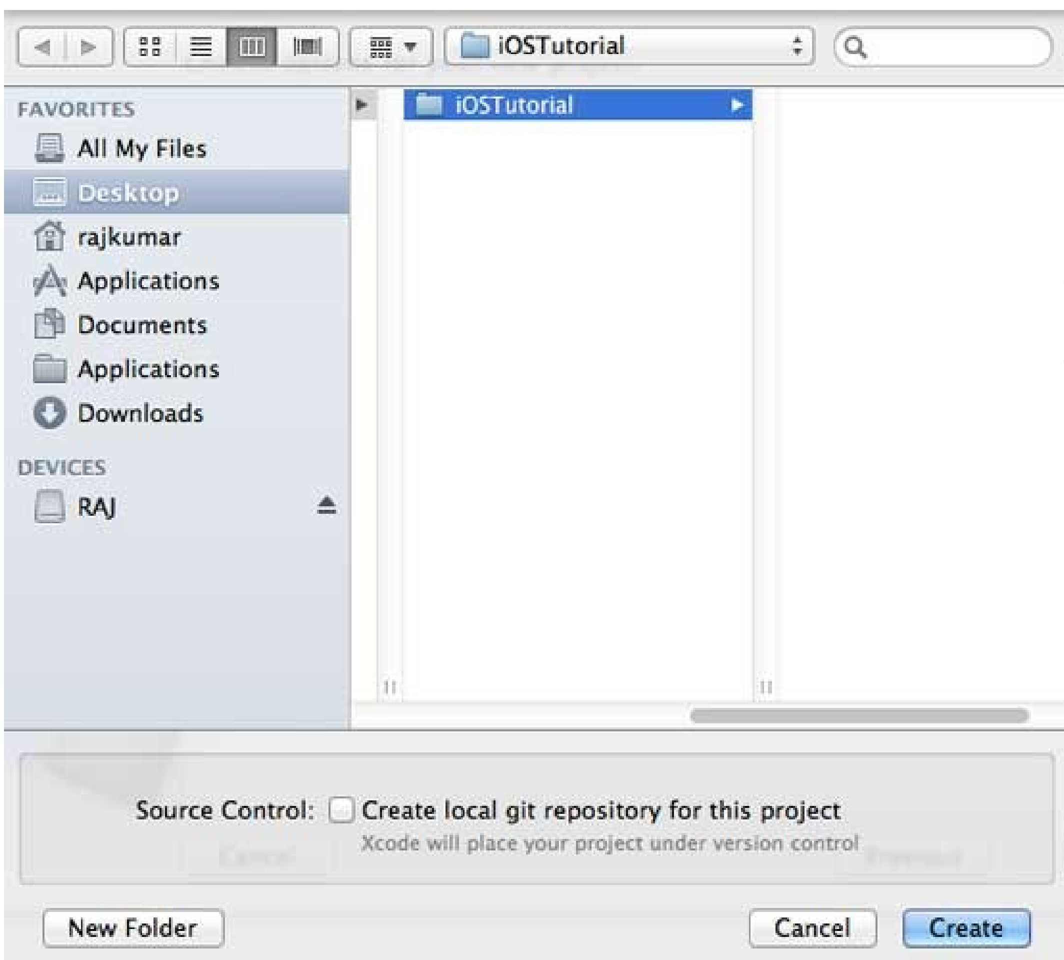


3. 接下来输入产品名称即应用程序名称、组织名称和公司标识符。

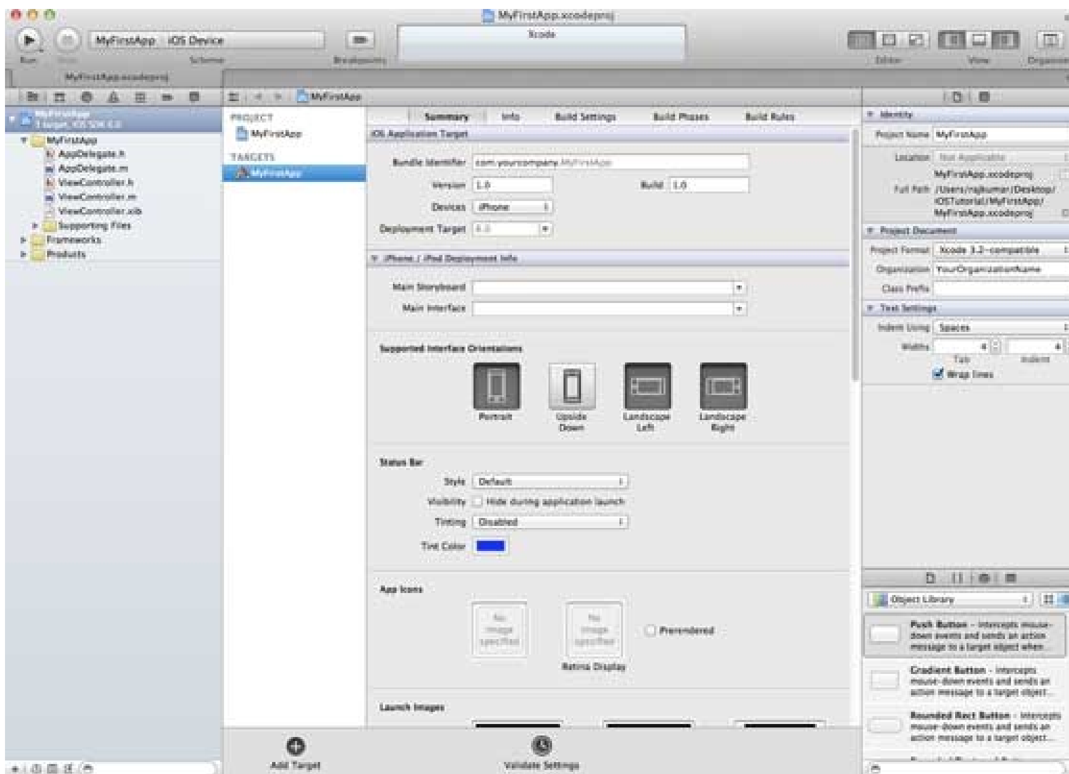


4. 确定已经选择自动应用计数，以自动释放超出范围的资源。单击下一步。

5.选择项目目录并选择创建

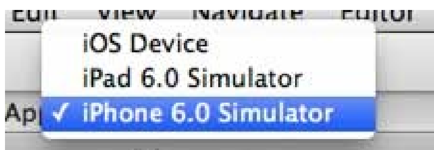


6. 你将看到如下所示的页面

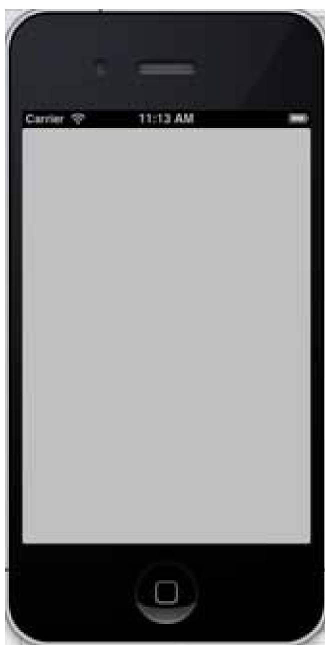


屏幕上方能够设置方向、生成和释放。有一个部署目标，设备支持4.3及以上版本的部署目标，这些不是必须的，现在只要专注于运行该应用程序。

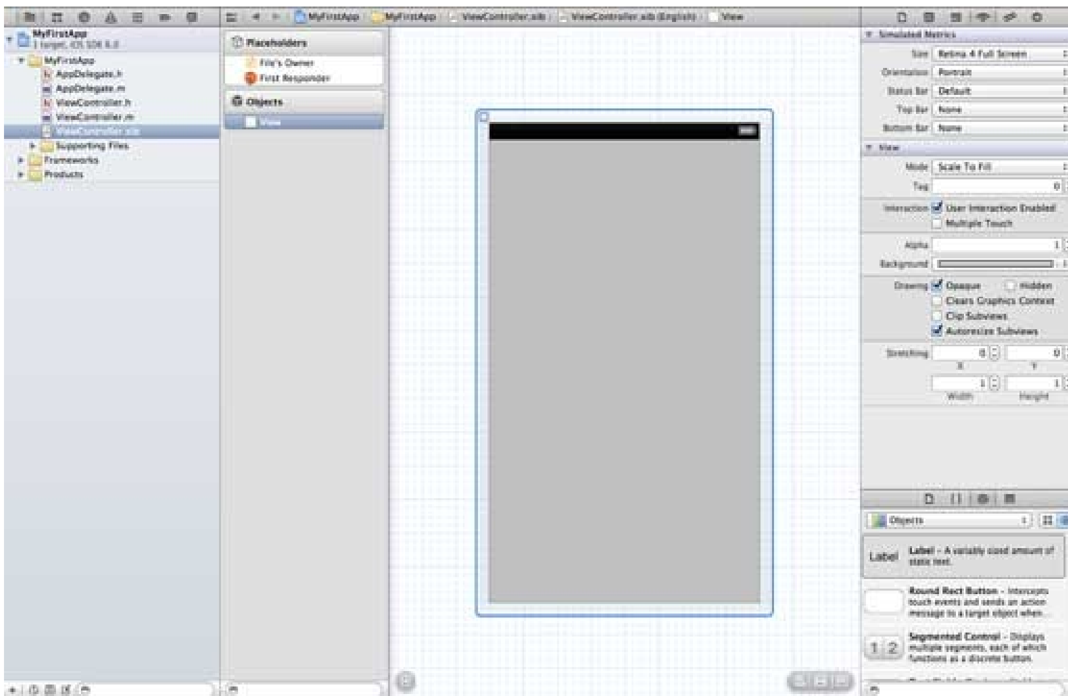
7. 在下拉菜单中选择iPhone Simulator并运行。



8. 成功运行第一个应用程序，将得到的输出，如下所示。



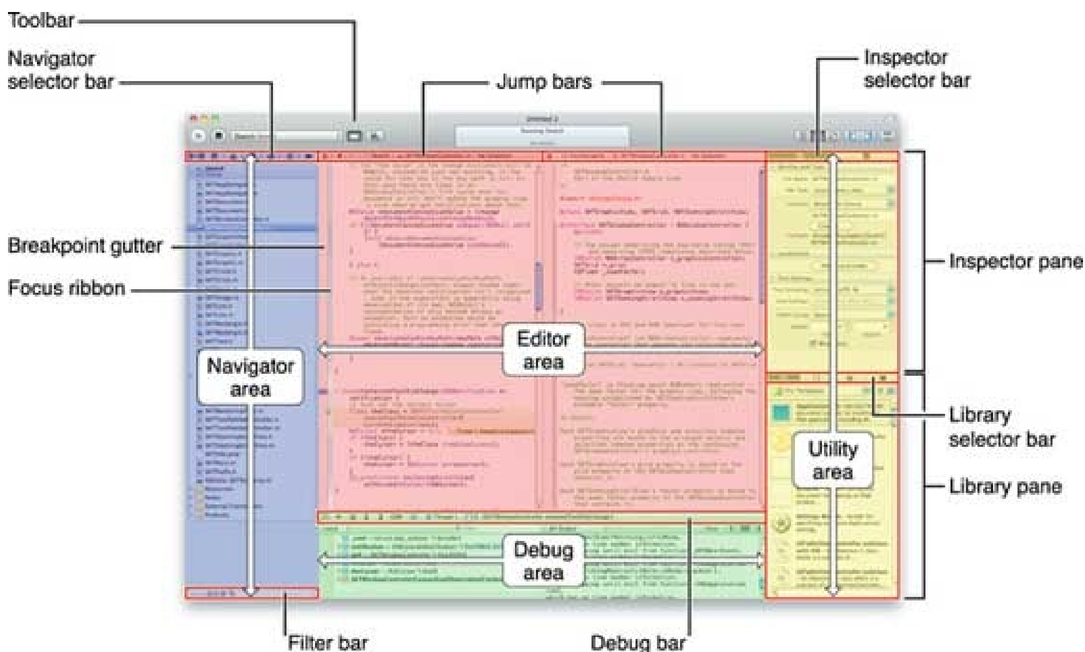
更改背景颜色使之有开始的界面生成器。选择ViewController.xib。在右侧选择背景选项，更改颜色并运行。



在上述项目中，默认情况下，部署目标已设置为iOS6.0且自动布局将被启用。

为确保应用程序能iOS4.3设备上正常运行，我们已经在开始创建应用程序时修改了部署目标，但我们不禁用自动布局，要取消自动布局，我们需要取消选择自动班上复选框在文件查看器的每个nib，也就是xib文件。

Xcode项目IDE的各部分显示如下（苹果Xcode4用户文档）



在上面所示的检查器选择器栏中可以找到文件检查器，且可以取消选择自动布局。当你想要的目标只有iOS6.0的设备时，可以使用自动布局。

当然，也可以使用新功能，如当加注到iOS6时，就可以使用passbook这一功能。现在，以Ios4.3作为部署目标。

深入了解第一款IOS应用程序代码

5个不同文件生成应用程序，如下所示

- AppDelegate.h
- AppDelegate.m
- ViewController.h
- ViewController.m
- ViewController.xib

我们使用单行注释（//）来解释简单代码，重要的项目代码解释在代码下方。

AppDelegate.h

```
// Header File that provides all UI related items.
#import &lt;UIKit/UIKit.h&gt;
 // Forward declaration (Used when class will be defined /imported
@class ViewController;

 // Interface for Appdelegate
@interface AppDelegate : UIResponder &lt;UIApplicationDelegate&gt;
// Property window
@property (strong, nonatomic) UIWindow *window;
 // Property Viewcontroller
@property (strong, nonatomic) ViewController *viewController;
//this marks end of interface
@end
```

代码说明

- AppDelegate调用UIResponder来处理Ios事件。
- 完成UIApplication的命令，提供关键应用程序事件，如启动完毕，终止，等等
- 在iOS设备的屏幕上用UIWindow对象来管理和协调各种视角，它就像其它加载视图的基本视图一样。通常一个应用程序只有一个窗口。
- UIViewController来处理屏幕流

AppDelegate.m

```
// Imports the class Appdelegate's interface
import "AppDelegate.h"

// Imports the viewcontroller to be loaded
#import "ViewController.h"
```

```
// Class definition starts here
@implementation AppDelegate

// Following method intimates us the application launched  successf
- (BOOL)application:(UIApplication *)application
 didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[UIWindow alloc] initWithFrame:
    [[UIScreen mainScreen] bounds]];
    // Override point for customization after application launch.
    self.viewController = [[ViewController alloc]
     initWithNibName:@"ViewController" bundle:nil];
    self.window.rootViewController = self.viewController;
    [self.window makeKeyAndVisible];
    return YES;
}

- (void)applicationWillResignActive:(UIApplication *)application
{
    /* Sent when the application is about to move from active to in
    This can occur for certain types of temporary interruptions
    (such as an incoming phone call or SMS message)
    or when the user quits the application and it begins the transi
    background state. Use this method to pause ongoing tasks, disab
    and throttle down OpenGL ES frame rates. Games should use this
    to pause the game.*/
}

- (void)applicationDidEnterBackground:(UIApplication *)application
{
    /* Use this method to release shared resources, save user data,
    timers, and store enough application state information    to re
    application to its current state in case it is terminated later
    application supports background execution, this method is calle
    of applicationWillTerminate: when the user quits.*/
}

- (void)applicationWillEnterForeground:(UIApplication *)applicatio
{
    /* Called as part of the transition from the background to the
    here you can undo many of the changes made on entering the back
}

- (void)applicationDidBecomeActive:(UIApplication *)application
{
    /* Restart any tasks that were paused (or not yet started) whil
    application was inactive. If the application was previously in
    optionally refresh the user interface.*/
}

- (void)applicationWillTerminate:(UIApplication *)application
{
```

```
    /* Called when the application is about to terminate. Save data
    See also applicationDidEnterBackground:. */
}

@end
```

代码说明

- 此处定义UIApplication。上面定义的所有方法都是应用程序UI调动和不包含任何用户定义的方法。
- UIWindow对象被分配用来保存应用程序分配对象。
- UIController作为窗口初始视图控制器
- 调用makeKeyAndVisible能使窗口可见

ViewController.h

```
#import

// Interface for class ViewController
@interface ViewController : UIViewController

@end
```

代码说明

- ViewController类继承UIViewController，为iOS应用程序提供基本视图管理模型。

ViewController.m

```
#import "ViewController.h"

// Category, an extension of ViewController class
@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically fr
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```

代码说明

- 在这里两种方法实现UIViewController类的基类中定义
- 初始视图加载后调用viewDidLoad中的安装程序
- 在内存警告的情况下调用didReceviveMemoryWarning

简介

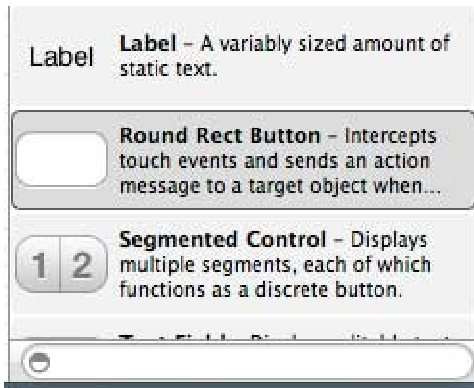
在iOS中，操作（action）和输出口（Outlet）指的是ibActions和ibOutlets，也就是ib接口生成器所在的地方。这些都和UI元素相关，我们将直观的了解他们后探讨如何实现他们。

步骤

1、让我们使用第一款iPhone应用程序。

2、从导航部分中的文件中选择ViewController.xib文件

3、从右手边得窗口下面显示的窗口格库中选择UI元素



4、拖拽UI元素到界面生成器的可视框中

5、添加标签和红色圆形按钮到可视图中



6、在工作区工具栏的右上角找到编辑器选择按钮，如下图所示

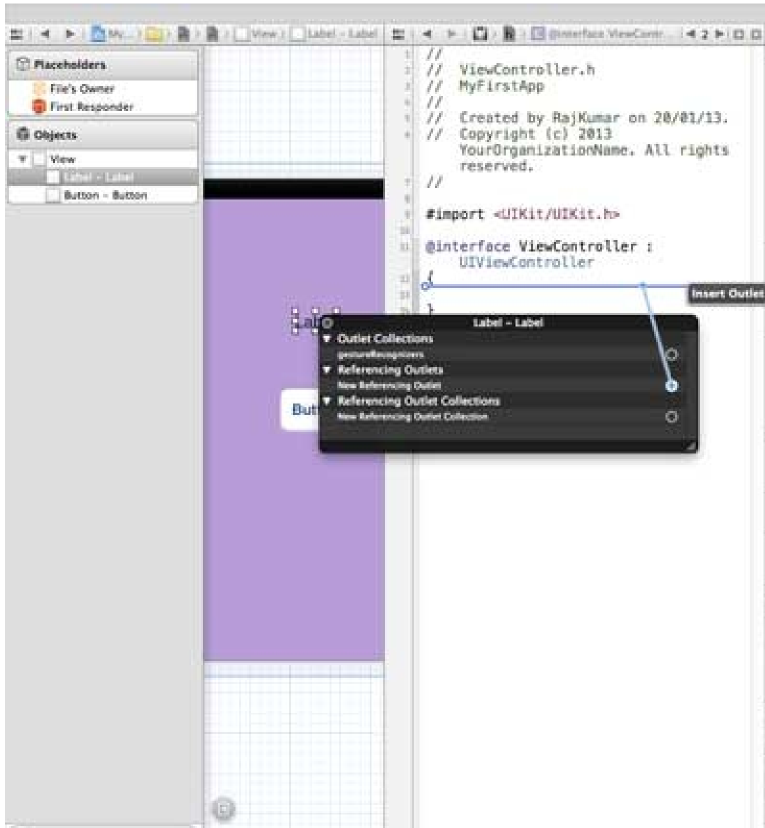


选择编辑器按钮

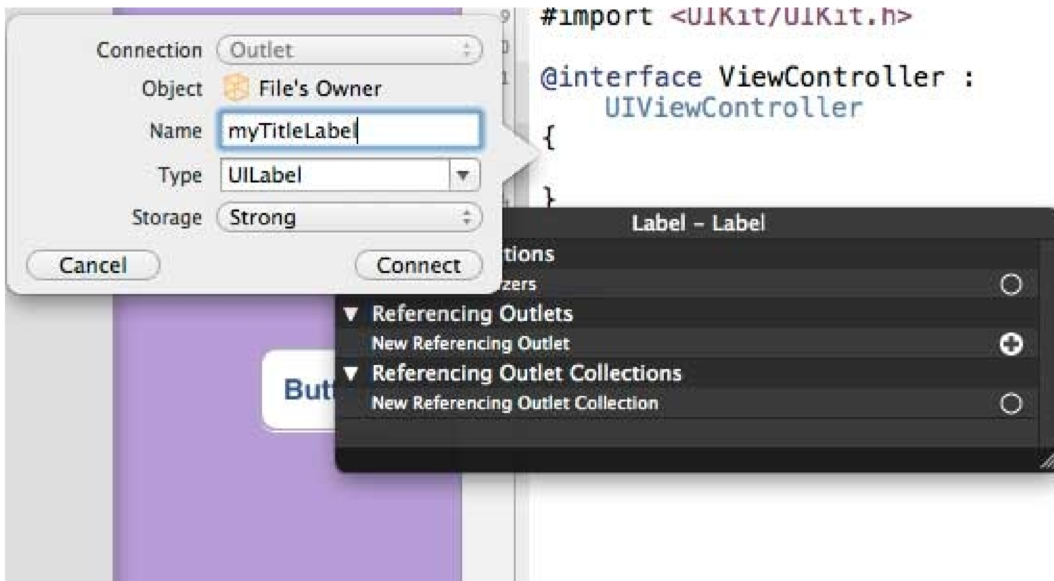


7、编辑器区域中心有两个窗口，ViewController.xib文件和ViewController.h

8、右击标签上的选择按钮，按住并拖动新引用参照，如下所示

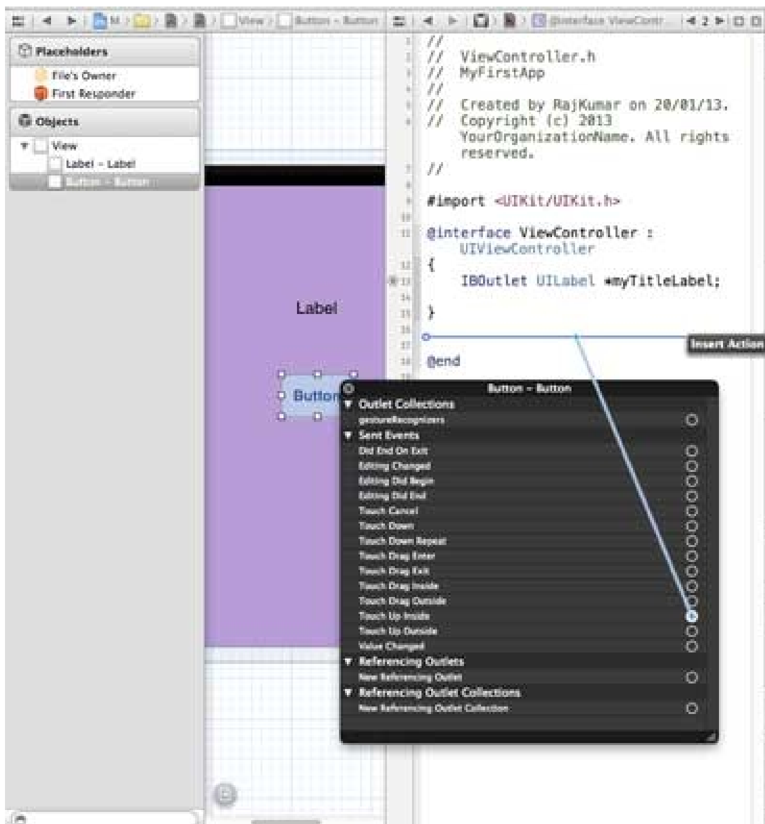


9、现在放在ViewController.h之间的大括号中。也可以放在文件中，如果是这样，必须在做这个之前已经添加了。如下所示

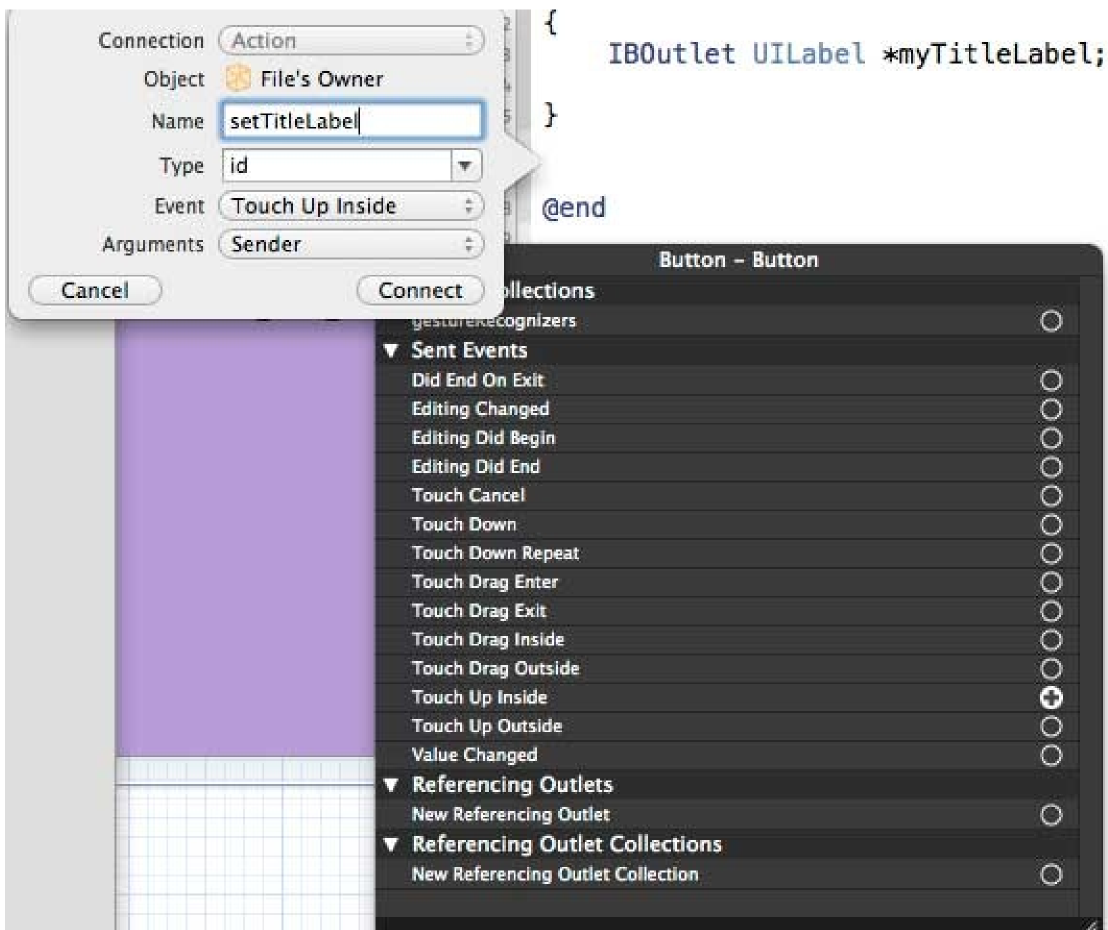


10. 输入输出口（Outlet）的标签名称，这里给出的是myTitleLable。单击链接，完成ibOutlet

11、同样的，添加操作，只需右击倒圆角矩形，选择触摸内心拖动它下方的大括号



12、重新命名为setTitleLable



13、 选择ViewController.m文件，有一种方法，如下所示

```
-(IBAction) setTitleLabel:(id)sender{
}
```

14、在上述的方法内，如下所示，添加一个语句

```
[myTitleLabel setTitleText:@"Hello"];
```

15、选择运行按钮运行该程序，得到如下的输出



16、单击按钮



17.、创建的参照（outlets）按钮标签已更改为对按钮执行的操作（actions）

18、由上可知，IBOutlet将创建对UIElement的引用（此处为UILable），同样的IBAction和UIButton通过执行操作和UIButton相链接。

19、当创建动作时通过选择不同的事件你可以做不同的操作。

委托（Delegates）示例

假设对象A调用B来执行一项操作，操作一旦完成，对象A就必须知道对象B已完成任务且对象A将执行其他必要操作。

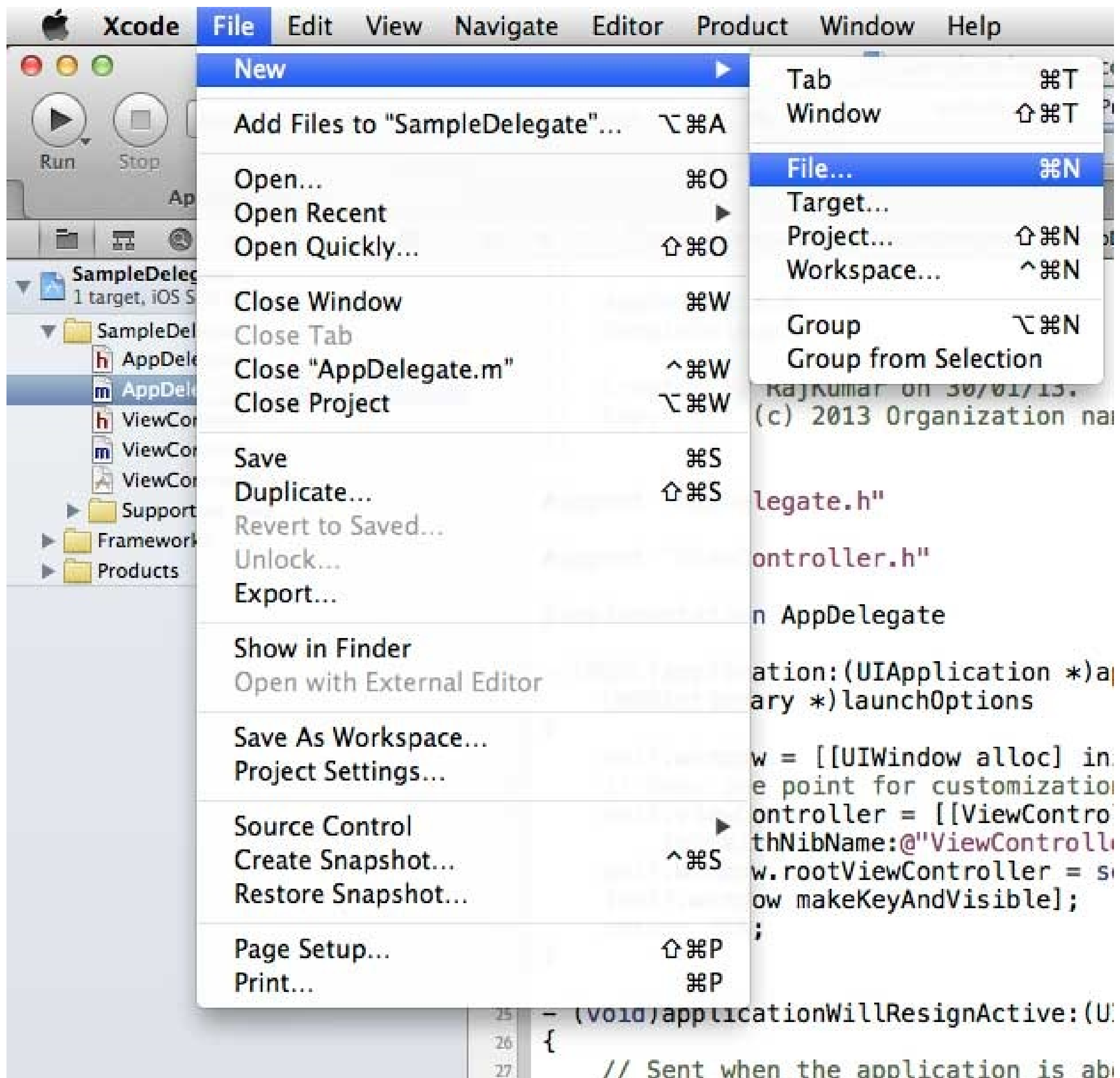
在上面的示例中的关键概念有

- A是B的委托对象
- B引用一个A
- A将实现B的委托方法
- B通过委托方法通知

创建一个委托（Delegates）对象

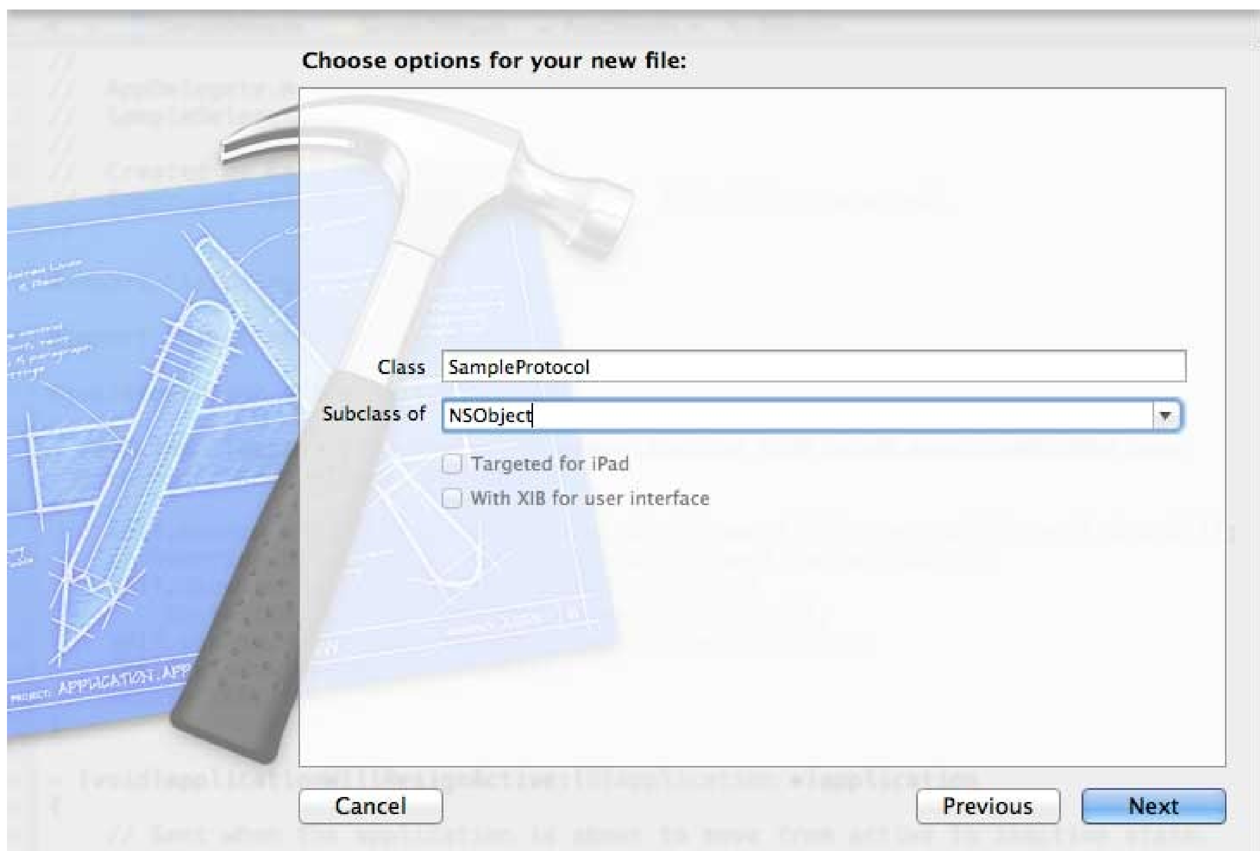
1. 创建一个单一视图的应用程序

2. 然后选择文件 File -> New -> File...



3. 然后选择Objective C单击下一步

4. 将SampleProtocol的子类命名为NSObject，如下所示



5. 然后选择创建

6.向SampleProtocol.h文件夹中添加一种协议，然后更新代码，如下所示：

```
#import &lt;Foundation/Foundation.h&gt;
// Protocol definition starts here
@protocol SampleProtocolDelegate &lt;NSObject&gt;
@required
- (void) processCompleted;
@end
// Protocol Definition ends here
@interface SampleProtocol : NSObject

{
   // Delegate to respond back
   id &lt;SampleProtocolDelegate&gt; _delegate;

}
@property (nonatomic,strong) id delegate;

-(void)startSampleProcess; // Instance method

@end
```

7. Implement the instance method by updating the SampleProtocol.m file as shown below.

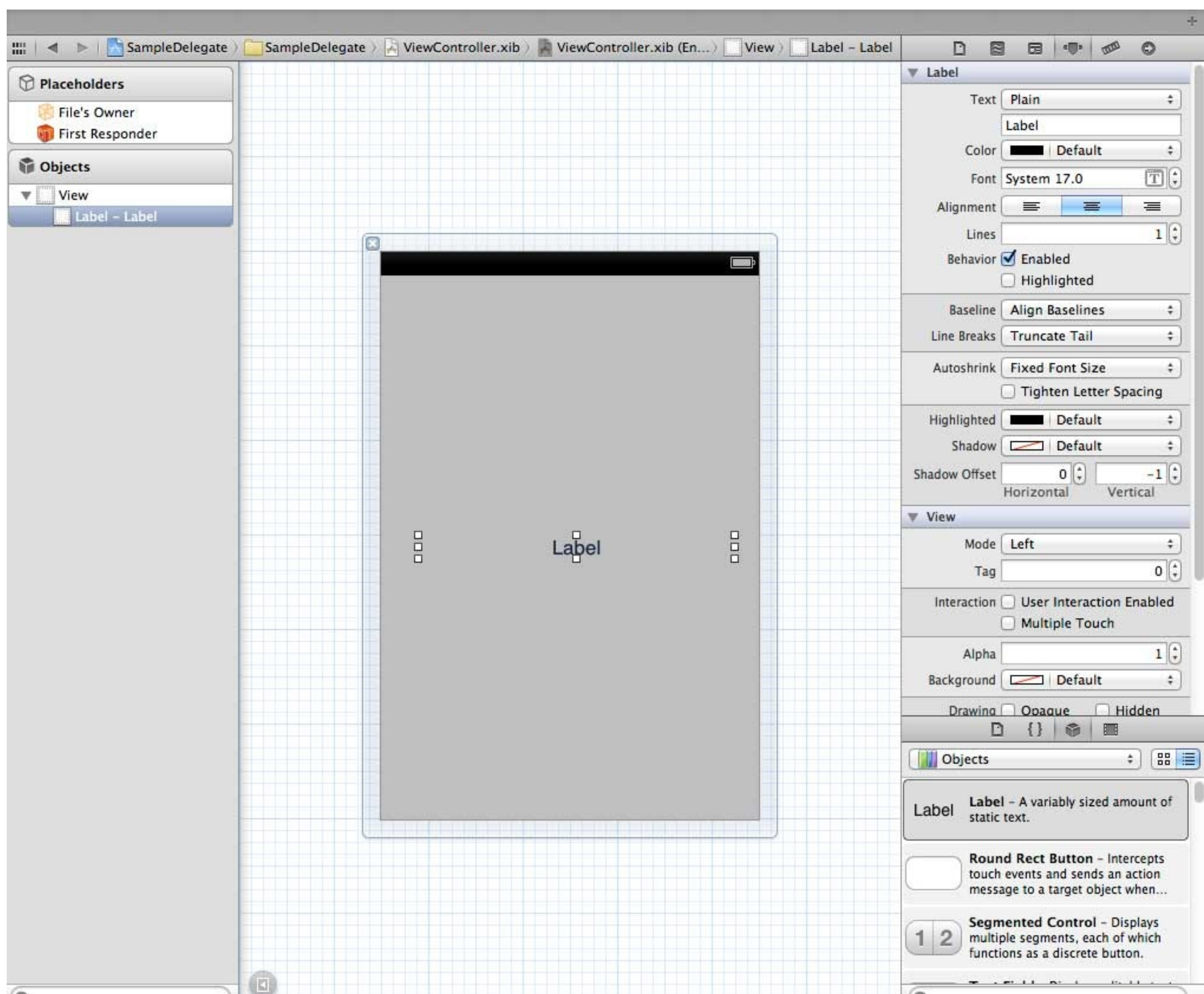
```
#import "SampleProtocol.h"

@implementation SampleProtocol

-(void)startSampleProcess{

    [NSTimer scheduledTimerWithTimeInterval:3.0 target:self.delegat
    selector:@selector(processCompleted) userInfo:nil repeats:NO];
}
@end
```

8. 将标签从对象库拖到UIView，从而在ViewController.xib中添加UILabel，如下所示:



9. 创建一个IBOutlet标签并命名为myLabel，然后按如下所示更新代码并在ViewController.h里显示SampleProtocolDelegate

```
#import &lt;UIKit/UIKit.h&gt;
#import "SampleProtocol.h"

@interface ViewController : UIViewController&lt;SampleProtocolDele
{
    IBOutlet UILabel *myLabel;
}
@end
```

10. 完成授权方法，为SampleProtocol创建对象和调用startSampleProcess方法。如下所示，更新ViewController.m文件

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    SampleProtocol *sampleProtocol = [[SampleProtocol alloc]init];
    sampleProtocol.delegate = self;
    [myLabel setText:@"Processing..."];
    [sampleProtocol startSampleProcess];
    // Do any additional setup after loading the view, typically fr
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

#pragma mark - Sample protocol delegate
-(void)processCompleted{
    [myLabel setText:@"Process Completed"];
}

@end
```

11. 将看到如下所示的输出结果，最初的标签也会继续运行，一旦授权方法被SampleProtocol对象所调用，标签运行程序的代码也会更新。



什么是UI元素？

UI元素是我们应用程序里可以看见的任何可视元素，其中一些元素响应用户的操作，如按钮、文本字段，有其他的丰富内容，如图像、标签等。

如何添加UI元素？

可以在界面生成器的参与下，在代码中添加UI元素。如果需要，我们可以使用他们其中之一。

我们关注的

通过代码，将集中于添加UI元素到应用程序。比较简单而直接的方法是使用界面生成器拖放UI元素。

方法

以下我们通过创建一款简单的IOS应用程序，来解释一些UI元素

步骤

1、在第一款IOS程序里一样，创建一个Viewbased应用程序

2、只更新ViewController.h和ViewController.m文件

3、然后我们将方法添加到ViewController.m文件中来创建UI元素

4、在viewDidLoad方法中调用此方法

5、重要的代码行已经在代码中通过在单行上方标注的方式进行了注释

用户界面元素列表

下面解释具体的UI元素和其相关的功能

具体的UI元素	功能
Text Fields-文本字段	用户界面元素，使用应用程序来获取用户输入
输入类型-TextFields	用户可以通过使用UITextField来赋予键盘输入属性
Buttons-按钮	用于处理用户操作
Label-标签	用于显示静态内容
Toolbar-工具栏	操纵当前视图所显示的东西
Status Bar-状态栏	显示设备的关键信息
Navigation Bar-导航栏	包含一个可以推断的视图控制器，并弹出导航控制器的导航按钮
Tab bar-选项卡栏	一般用于各个子任务、视图或同一视图中的模型之间的切换.
Image View-图像视图	用于显示一个简单的图像序列
Scroll View-滚动视图	用来显示更多屏幕区域的内容
Table View-列表视图	用于在多个行或部分中显示可滚动列表的数据
IOS分割视图(Split View)	用于在详细信息窗格上显示两个窗格与主窗格的控制信息
Text View-文本视图	用于显示滚动列表的文本信息可以被选中和编辑
View Transition -视图切换	各种视图查看之间的切换
Pickers-选择器	用来显示从列表中选择一个特定的数据
Switches-开关	用作禁用和启用操作
IOS滑块(Sliders)	用来允许用户在允许的值范围内选对一个值
IOS警告对话框(Alerts)	用来给用户提供重要的信息
IOS图标(Icons)	它是图像，表示用于行动或描绘与应用程序相关的东西

文本字段的使用

文本字段是一个用户界面元素，通过应用程序来获取用户输入。

一个UITextfield如下所示：

重要的文本字段的属性

- 在没有任何用户输入时，显示占位符
- 正常文本
- 自动更正型
- 键盘类型
- 返回键类型
- 清除按钮模式
- 对齐方式
- 委托

更新xib中的属性

可以在Utility area（实用区域，窗口的右侧）更改xib在属性查看器中的文本字段属性。



文本字段委托

我们可以通过右击 UIElement 界面生成器中设置委托并将它连接到文件的所有者，如下所示：



使用委托的步骤：

- 1.设置委托如上图所示
- 2.添加委托到您的响应类
- 3.执行文本字段代表，重要的文本字段代表如下:

```
- (void)textFieldDidBeginEditing:(UITextField *)textField
```

```
- (void)textFieldDidEndEditing:(UITextField *)textField
```

- 4.正如其名称所暗示，上述两个委托分别叫做编辑的文本字段和结束编辑
- 5. 其他的委托请查看 UITextDelegate Protocol 参考手册。

实例

以下我们使用简单的实例来创建UI元素

ViewController 类将采用UITextFieldDelegate，修改ViewController.h文件，如下所示:

将方法addTextField添加到我们的 ViewController.m 文件

然后在 viewDidLoad 方法中调用此方法

在ViewController.m中更新viewDidLoad，如下所示

```
#import "ViewController.h"
@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    //The custom method to create our textfield is called
    [self addTextField];
    // Do any additional setup after loading the view, typically fr
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

-(void)addTextField{
   // This allocates a label
   UILabel *prefixLabel = [[UILabel alloc]initWithFrame:CGRectZero]
   //This sets the label text
   prefixLabel.text =@"## ";
   // This sets the font for the label
   [prefixLabel setFont:[UIFont boldSystemFontOfSize:14]];
   // This fits the frame to size of the text
   [prefixLabel sizeToFit];

   // This allocates the textfield and sets its frame
   UITextField *textField = [[UITextField  alloc] initWithFrame:
   CGRectMake(20, 50, 280, 30)];

   // This sets the border style of the text field
   textField.borderStyle = UITextBorderStyleRoundedRect;
   textField.contentVerticalAlignment =
   UIControlContentVerticalAlignmentCenter;
   [textField setFont:[UIFont boldSystemFontOfSize:12]];

   //Placeholder text is displayed when no text is typed
   textField.placeholder = @"Simple Text field";

   //Prefix label is set as left view and the text starts after tha
   textField.leftView = prefixLabel;
```

```
    //It set when the left prefixLabel to be displayed
    textField.leftViewMode = UITextFieldViewModeAlways;

    // Adds the textField to the view.
    [self.view addSubview:textField];

    // sets the delegate to the current class
    textField.delegate = self;
}

// pragma mark is used for easy access of code in Xcode
#pragma mark - TextField Delegates

// This method is called once we click inside the textField
-(void)textFieldDidBeginEditing:(UITextField *)textField{
    NSLog(@"Text field did begin editing");
}

// This method is called once we complete editing
-(void)textFieldDidEndEditing:(UITextField *)textField{
    NSLog(@"Text field ended editing");
}

// This method enables or disables the processing of return key
-(BOOL) textFieldShouldReturn:(UITextField *)textField{
     [textField resignFirstResponder];
     return YES;
}

- (void)viewDidUnload {
    label = nil;
    [super viewDidUnload];
}

@end
```

运行该应用程序会看到下面的输出

委托调用的方法基于用户操作。要知道调用委托时请参阅控制台输出。

为什么使用不同的输入类型？

键盘输入的类型帮助我们从用户获取必需的输入。

它移除不需要的键，并包括所需的部分。用户可以通过使用 UITextField 的键盘属性设置输入的类型。

- 如：文本字段（textField）。 keyboardType = UIKeyboardTypeDefault

键盘输入类型

输入的类型	描述
UIKeyboardTypeASCIICapable	键盘包括所有标准的 ASCII 字符。
UIKeyboardTypeNumbersAndPunctuation	键盘显示数字和标点。
UIKeyboardTypeURL	键盘的 URL 项优化。
UIKeyboardTypeNumberPad	键盘用于 PIN 输入和显示一个数字键盘。
UIKeyboardTypePhonePad	键盘对输入电话号码进行了优化。
UIKeyboardTypeNamePhonePad	键盘用于输入姓名或电话号码。
UIKeyboardTypeEmailAddress	键盘对输入电子邮件地址的优化。
UIKeyboardTypeDecimalPad	键盘用来输入十进制数字。
UIKeyboardTypeTwitter	键盘对 twitter @ 和 # 符号进行了优化。

添加自定义方法 addTextFieldWithDifferentKeyboard

```
-(void) addTextFieldWithDifferentKeyboard{

    UITextField *textField1= [[UITextField alloc]initWithFrame:
    CGRectMake(20, 50, 280, 30)];
    textField1.delegate = self;
    textField1.borderStyle = UITextBorderStyleRoundedRect;
    textField1.placeholder = @"Default Keyboard";
    [self.view addSubview:textField1];

    UITextField *textField2 = [[UITextField alloc]initWithFrame:
    CGRectMake(20, 100, 280, 30)];
    textField2.delegate = self;
    textField2.borderStyle = UITextBorderStyleRoundedRect;
    textField2.keyboardType = UIKeyboardTypeASCIICapable;
    textField2.placeholder = @"ASCII keyboard";
    [self.view addSubview:textField2];

    UITextField *textField3 = [[UITextField alloc]initWithFrame:
    CGRectMake(20, 150, 280, 30)];
    textField3.delegate = self;
    textField3.borderStyle = UITextBorderStyleRoundedRect;
    textField3.keyboardType = UIKeyboardTypePhonePad;
    textField3.placeholder = @"Phone pad keyboard";
    [self.view addSubview:textField3];

    UITextField *textField4 = [[UITextField alloc]initWithFrame:
    CGRectMake(20, 200, 280, 30)];
    textField4.delegate = self;
    textField4.borderStyle = UITextBorderStyleRoundedRect;
    textField4.keyboardType = UIKeyboardTypeDecimalPad;
    textField4.placeholder = @"Decimal pad keyboard";
    [self.view addSubview:textField4];

    UITextField *textField5= [[UITextField alloc]initWithFrame:
    CGRectMake(20, 250, 280, 30)];
    textField5.delegate = self;
    textField5.borderStyle = UITextBorderStyleRoundedRect;
    textField5.keyboardType = UIKeyboardTypeEmailAddress;
    textField5.placeholder = @"Email keyboard";
    [self.view addSubview:textField5];

    UITextField *textField6= [[UITextField alloc]initWithFrame:
    CGRectMake(20, 300, 280, 30)];
    textField6.delegate = self;
    textField6.borderStyle = UITextBorderStyleRoundedRect;
    textField6.keyboardType = UIKeyboardTypeURL;
    textField6.placeholder = @"URL keyboard";
    [self.view addSubview:textField6];
}
```

在 ViewController.m 中更新 viewDidLoad，如下所示

```
(void)viewDidLoad
{
   [super viewDidLoad];
   //The custom method to create textfield with different keyboard
   [self addTextFieldWithDifferentKeyboard];
   //Do any additional setup after loading the view, typically fror
}
```

输出

现在当我们运行应用程序时我们就会得到下面的输出:

选择不同的文本区域我们将看到不同的键盘。

按钮使用

按钮用于处理用户操作。它截取触摸事件，并将消息发送到目标对象。

圆角矩形按钮

在 xib 中的按钮属性

您可以在Utility area（实用区域，窗口的右侧）的属性检查器的更改 xib 按钮属性。

按钮类型

- UIButtonTypeCustom
- UIButtonTypeRoundedRect
- UIButtonTypeDetailDisclosure
- UIButtonTypeInfoLight
- UIButtonTypeInfoDark
- UIButtonTypeContactAdd

重要的属性

- imageView
- titleLabel

重要的方法

```
+ (id)buttonWithType:(UIButtonType)buttonType
```

```
- (UIImage *)backgroundImageForState:(UIControlState)state
```

```
- (UIImage *)imageForState:(UIControlState)state
```

```
- (void)setTitle:(NSString *)title forState:(UIControlState)state
```

```
- (void)addTarget:(id)target action:(SEL)action forControlEvents: (
```

添加自定义方法 addDifferentTypesOfButton

```
-(void)addDifferentTypesOfButton
{
    // A rounded Rect button created by using class method
    UIButton *roundRectButton = [UIButton buttonWithType:
    UIButtonTypeRoundedRect];
    [roundRectButton setFrame:CGRectMake(60, 50, 200, 40)];
    // sets title for the button
    [roundRectButton setTitle:@"Rounded Rect Button" forState:
    UIControlStateNormal];
    [self.view addSubview:roundRectButton];

    UIButton *customButton = [UIButton buttonWithType: UIButtonType
    [customButton setBackgroundColor: [UIColor lightGrayColor]];
    [customButton setTitleColor:[UIColor blackColor] forState:
    UIControlStateHighlighted];
    //sets background image for normal state
    [customButton setBackgroundImage:[UIImage imageNamed:
    @"Button_Default.png"]
    forState:UIControlStateNormal];
    //sets background image for highlighted state
    [customButton setBackgroundImage:[UIImage imageNamed:
    @"Button_Highlighted.png"]
    forState:UIControlStateHighlighted];
    [customButton setFrame:CGRectMake(60, 100, 200, 40)];
    [customButton setTitle:@"Custom Button" forState:UIControlState
    [self.view addSubview:customButton];

    UIButton *detailDisclosureButton = [UIButton buttonWithType:
    UIButtonTypeDetailDisclosure];
    [detailDisclosureButton setFrame:CGRectMake(60, 150, 200, 40)];
    [detailDisclosureButton setTitle:@"Detail disclosure" forState
    UIControlStateNormal];
    [self.view addSubview:detailDisclosureButton];

    UIButton *contactButton = [UIButton buttonWithType:
    UIButtonTypeContactAdd];
    [contactButton setFrame:CGRectMake(60, 200, 200, 40)];
    [self.view addSubview:contactButton];

    UIButton *infoDarkButton = [UIButton buttonWithType:
    UIButtonTypeInfoDark];
    [infoDarkButton setFrame:CGRectMake(60, 250, 200, 40)];
    [self.view addSubview:infoDarkButton];

    UIButton *infoLightButton = [UIButton buttonWithType:
    UIButtonTypeInfoLight];
    [infoLightButton setFrame:CGRectMake(60, 300, 200, 40)];
    [self.view addSubview:infoLightButton];
}
```

注意：

我们将命名为"Button_Default.png"和"Button_Highlighted.png"的个图像添加到我们的项目，可以通过将图像拖到列出了我们的项目文件的导航区域来完成。

在 ViewController.m 中更新 viewDidLoad，如下所示

```
(void)viewDidLoad
{
    [super viewDidLoad];
    //The custom method to create our different types of button is
    [self addDifferentTypesOfButton];
    //Do any additional setup after loading the view, typically fro
}
```

输出

现在当我们运行应用程序时我们就会得到下面的输出：



标签的使用

标签用于显示静态内容，包括单独的一行或多行。

重要的属性

- textAlignment
- textColor
- text
- numberOflines
- lineBreakMode

添加自定义方法 addLabel

```
-(void)addLabel{
    UILabel *aLabel = [[UILabel alloc]initWithFrame:
    CGRectMake(20, 200, 280, 80)];
    aLabel.numberOfLines = 0;
    aLabel.textColor = [UIColor blueColor];
    aLabel.backgroundColor = [UIColor clearColor];
    aLabel.textAlignment = UITextAlignmentCenter;
    aLabel.text = @"This is a sample text\n of multiple lines.
    here number of lines is not limited.";
    [self.view addSubview:aLabel];
}
```

在 ViewController.m 中更新 viewDidLoad，如下所示:

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    //The custom method to create our label is called
    [self addLabel];
    // Do any additional setup after loading the view, typically fr
}
```

输出

运行应用程序，就会得到下面的输出:



工具栏的使用

我们可以使用工具栏修改视图元素。

如，邮件应用程序里的收件箱栏中有删除、分享、答复等等。如下所示：



重要的属性

- barStyle
- items

添加自定义方法 addToolbar

```
-(void)addToolbar
{
    UIBarButtonItem *spaceItem = [[UIBarButtonItem alloc]
    initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace
    target:nil action:nil];
    UIBarButtonItem *customItem1 = [[UIBarButtonItem alloc]
    initWithTitle:@"Tool1" style:UIBarButtonItemStyleBordered
    target:self action:@selector(toolBarItem1:)];
    UIBarButtonItem *customItem2 = [[UIBarButtonItem alloc]
    initWithTitle:@"Tool2" style:UIBarButtonItemStyleDone
    target:self action:@selector(toolBarItem2:)];
    NSArray *toolbarItems = [NSArray arrayWithObjects:
    customItem1,spaceItem, customItem2, nil];
    UIToolbar *toolbar = [[UIToolbar alloc]initWithFrame:
    CGRectMake(0, 366+54, 320, 50)];
    [toolbar setBarStyle:UIBarStyleBlackOpaque];
    [self.view addSubview:toolbar];
    [toolbar setItems:toolbarItems];
}
```

为了解所执行的操作我们在我们的ViewController.xib中添加UILabel Iboutlet并为UILabel 创建命名为标签的IBoutlet。

我们还需要添加两个方法来执行，如下所示的工具栏项的操作:

```
-(IBAction)toolBarItem1:(id)sender{
    [label setText:@"Tool 1 Selected"];
}

-(IBAction)toolBarItem2:(id)sender{
    [label setText:@"Tool 2 Selected"];
}
```

在ViewController.m中更新 viewDidLoad，如下所示:

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // The method hideStatusbar called after 2 seconds
    [self addToolbar];
    // Do any additional setup after loading the view, typically fr
}
```

输出

现在当我们运行该应用程序我们会看到下面的输出。



单击我们得到的 tool1 和 tool2 栏按钮



状态栏的使用

状态栏显示设备的关键信息。

- 设备模型或网络提供商
- 网络信号强度
- 电池使用量
- 时间

状态栏如下所示：



隐藏状态栏的方法

```
[[UIApplication sharedApplication] setStatusBarHidden:YES];
```

另一种隐藏状态栏的方法

我们还可以通过添加行，并在info.plist 的帮助下选择 UIStatusBarHidden 隐藏状态栏，并使其值为否（NO）。

在类中添加自定义方法 hideStatusbar

它隐藏状态栏进行动画处理，并也调整我们认为占据状态栏空间的大小。

```
-(void)hideStatusbar{
    [[UIApplication sharedApplication] setStatusBarHidden:YES
    withAnimation:UIStatusBarAnimationFade];
    [UIView beginAnimations:@"Statusbar hide" context:nil];
    [UIView setAnimationDuration:0.5];
    [self.view setFrame:CGRectMake(0, 0, 320, 480)];
    [UIView commitAnimations];
}
```

在 ViewController.m 中更新 viewDidLoad，如下所示：

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    // The method hideStatusbar called after 2 seconds
    [self performSelector:@selector(hideStatusbar)
    withObject:nil afterDelay:2.0];
    // Do any additional setup after loading the view, typically fr
}
```

初始输出以及2秒后输出

IOS导航栏的使用

导航栏包含导航控制器的导航的按钮。在导航栏中的标题是当前视图控制器的标题。

示例代码和步骤

1.创视图应用程序

1. 现在，选择应用程序 Delegate.h， 添加导航控制器的属性，如下所示:

```
#import <UIKit/UIKit.h>

@class ViewController;

@interface AppDelegate : UIResponder <UIApplicationDelegate>

@property (strong, nonatomic) UIWindow *window;

@property (strong, nonatomic) ViewController *viewController;

@property (strong, nonatomic) UINavigationController *navControlle

@end
```

3. 更新应用程序: didFinishLaunchingWithOptions:方法，在AppDelegate.m文件分配的导航控制器，并使其成为窗口的根视图控制器，如下所示：

```
- (BOOL)application:(UIApplication *)application didFinishLaunchin
{
    self.window = [[UIWindow alloc] initWithFrame:
    [[UIScreen mainScreen] bounds]];
    // Override point for customization after application launch.
    self.viewController = [[ViewController alloc]
    initWithNibName:@"ViewController" bundle:nil];
    //Navigation controller init with ViewController as root
    UINavigationController *navController = [[UINavigationControlle
    initWithRootViewController:self.viewController];
    self.window.rootViewController = navController;
    [self.window makeKeyAndVisible];
    return YES;
}
```

4.现在，通过选择**File** -> **New** ->**File...** -> Objective C Class 添加新的类文件 TempViewController，然后将类命名 TempViewController 与 UIViewController 的子类。

5.在ViewController.h中添加navButon，如下所示

```
// ViewController.h
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController
{
    UIButton *navButton;
}
@end
```

6.现在添加方法addNavigationBarItem并在viewDidLoad调用方法

7. 为导航项创建方法

1. 我们还需要创建另一种方法到另一视图控制器 TempViewController。
2. 更新后的ViewController.m，如下所示:

```
// ViewController.m
#import "ViewController.h"
#import "TempViewController.h"
@interface ViewController ()

@end
@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self addNavigationBarButton];
    //Do any additional setup after loading the view, typically fro
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

-(IBAction)pushNewView:(id)sender{
   TempViewController *tempVC =[[TempViewController alloc]
   initWithNibName:@"TempViewController" bundle:nil];
   [self.navigationController pushViewController:tempVC animated:YE
}

-(IBAction)myButtonClicked:(id)sender{
   // toggle hidden state for navButton
   [navButton setHidden:!nav.hidden];
}

-(void)addNavigationBarButton{
   UIBarButtonItem *myNavBtn = [[UIBarButtonItem alloc] initWithTit
   @"MyButton" style:UIBarButtonItemStyleBordered target:
   self action:@selector(myButtonClicked:)];

   [self.navigationController.navigationBar setBarStyle:UIBarStyleB
   [self.navigationItem setRightBarButtonItem:myNavBtn];

   // create a navigation push button that is initially hidden
   navButton = [UIButton buttonWithType:UIButtonTypeRoundedRect];
   [navButton setFrame:CGRectMake(60, 50, 200, 40)];
   [navButton setTitle:@"Push Navigation" forState:UIControlStateNo
   [navButton addTarget:self action:@selector(pushNewView:)
   forControlEvents:UIControlEventTouchUpInside];
   [self.view addSubview:navButton];
   [navButton setHidden:YES];
}
@end
```

1. 现在当我们运行应用程序时我们就会得到下面的输出



1. 单击 MyButton 导航按钮，切换导航按钮可见性
2. 单击导航按钮，显示另一个视图控制器，如下所示



IOS选项卡栏的使用

它一般用于在同一视图中各个子任务、 视图或的模型之间切换。

选项卡栏的示例如下所示：



重要的属性

- backgroundImage
- items
- selectedItem

示例代码和步骤

1. 创建一个新的项目，选择 **Tabbed Application** 替代视图应用程序 ,点击下一步, 输入项目名称和选择 **create**.

1. 这里默认创建两个视图控制器和标签栏添加到我们的应用程序。

3. **AppDelegate.m didFinishLaunchingWithOptions**方法如下：

```
- (BOOL)application:(UIApplication *)application didFinishLaunchin
{
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainSc
    bounds]];
    // Override point for customization after application launch.
    UIViewController *viewController1 = [[FirstViewController alloc
    initWithNibName:@"FirstViewController" bundle:nil];
    UIViewController *viewController2 = [[SecondViewController allo
    initWithNibName:@"SecondViewController" bundle:nil];
    self.tabBarController = [[UITabBarController alloc] init];
    self.tabBarController.viewControllers = @[viewController1,
    viewController2];
    self.window.rootViewController = self.tabBarController;
    [self.window makeKeyAndVisible];
    return YES;
}
```

1. 两个视图控制器被用来分配作为选项卡栏控制器的视图控制器
2. 运行应用程序,得到如下结果：



图像视图的使用

图像视图用于显示单个图像或动画序列的图像。

重要的属性

- image
- highlightedImage
- userInteractionEnabled
- animationImages
- animationRepeatCount

重要的方法

```
- (id)initWithImage:(UIImage *)image
```

```
- (id)initWithImage:(UIImage *)image highlightedImage:
  (UIImage *)highlightedImage
```

```
- (void)startAnimating
```

```
- (void)stopAnimating
```

添加自定义方法 addImageView

```
-(void)addImageView{
    UIImageView *imgview = [[UIImageView alloc]
    initWithFrame:CGRectMake(10, 10, 300, 400)];
    [imgview setImage:[UIImage imageNamed:@"AppleUSA1.jpg"]];
    [imgview setContentMode:UIViewContentModeScaleAspectFit];
    [self.view addSubview:imgview];
}
```

添加另一个自定义方法 addImageViewWithAnimation

这种方法解释了如何对imageView 中的图像进行动画处理

```
-(void)addImageViewWithAnimation{
    UIImageView *imgview = [[UIImageView alloc]
    initWithFrame:CGRectMake(10, 10, 300, 400)];
    // set an animation
    imgview.animationImages = [NSArray arrayWithObjects:
    [UIImage imageNamed:@"AppleUSA1.jpg"],
    [UIImage imageNamed:@"AppleUSA2.jpg"], nil];
    imgview.animationDuration = 4.0;
    imgview.contentMode = UIViewContentModeCenter;
    [imgview startAnimating];
    [self.view addSubview:imgview];
}
```

注意： 我们必须添加命名为"AppleUSA1.jpg"和"AppleUSA2.jpg"到我们的项目，可以通过将图像拖到我们导航区域，其中列出了我们的项目文件所做的图像。

在 ViewController.m 中更新 viewDidLoad，如下所示

```
(void)viewDidLoad
{
   [super viewDidLoad];
   [self addImageView];
}
```

滚动视图的使用

如果内容超出屏幕的大小就会使用到滚动视图来显示隐藏的部分。

它可以包含所有的其他用户界面元素 如图像视图、 标签、 文本视图甚至另一个滚动视图。

重要的属性

- contentSize
- contentInset
- contentOffset
- delegate

重要的方法

```
- (void)scrollRectToVisible:(CGRect)rect animated:(BOOL)animated
```

```
- (void)setContentOffset:(CGPoint)contentOffset animated:(BOOL)anir
```

重要的委托方法

在ViewController.h中，加入<uiscrollviewdelegate>滚动视图和声明滚动视图让类符合委托协议，如下所示:</uiscrollviewdelegate>

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController<UIScrollViewDelegate>
{
    UIScrollView *myScrollView;
}

@end
```

添加自定义方法 addScrollView

```
-(void)addScrollView{
    myScrollView = [[UIScrollView alloc]initWithFrame:
    CGRectMake(20, 20, 280, 420)];
    myScrollView.accessibilityActivationPoint = CGPointMake(100, 1(
    imgView = [[UIImageView alloc]initWithImage:
    [UIImage imageNamed:@"AppleUSA.jpg"]];
    [myScrollView addSubview:imgView];
    myScrollView.minimumZoomScale = 0.5;
    myScrollView.maximumZoomScale = 3;
    myScrollView.contentSize = CGSizeMake(imgView.frame.size.width,
    imgView.frame.size.height);
    myScrollView.delegate = self;
    [self.view addSubview:myScrollView];
}
```

注意： 我们必须添加一个命名为"AppleUSA1.jpg"到我们的项目，可以通过将图像拖到我们导航区域，其中列出了我们的项目文件所做的图像。图像应高于设备的高度。

ViewController.m中实现scrollView 委托

```
 -(UIView *)viewForZoomingInScrollView:(UIScrollView *)scrollView{
    return imgView;
}
-(void)scrollViewDidEndDecelerating:(UIScrollView *)scrollView{
    NSLog(@"Did end decelerating");
}
-(void)scrollViewDidScroll:(UIScrollView *)scrollView{
//    NSLog(@"Did scroll");
}
-(void)scrollViewDidEndDragging:(UIScrollView *)scrollView
  willDecelerate:(BOOL)decelerate{
    NSLog(@"Did end dragging");
}
-(void)scrollViewWillBeginDecelerating:(UIScrollView *)scrollView{
    NSLog(@"Did begin decelerating");
}
-(void)scrollViewWillBeginDragging:(UIScrollView *)scrollView{
    NSLog(@"Did begin dragging");
}
```

在 **ViewController.m** 中更新 **viewDidLoad**，如下所示

```
(void)viewDidLoad
{
   [super viewDidLoad];
   [self addScrollView];
   //Do any additional setup after loading the view, typically fror
}
```

输出

现在当我们运行该应用程序我们会看到下面的输出。一旦滚动滚动视图，将能够查看图像的其余部分:



表格视图的使用

IOS表格视图由单元格 （一般可重复使用） 组成，用于显示垂直滚动的视图。

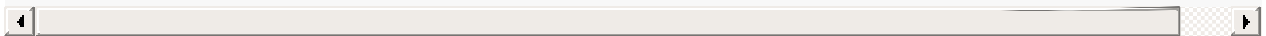
在iOS 中,表格视图用于显示数据列表,如联系人、待办事项或购物项列表。

重要的属性

- delegate
- dataSource
- rowHeight
- sectionFooterHeight
- sectionHeaderHeight
- separatorColor
- tableHeaderView
- tableFooterView

重要的方法

```
- (UITableViewCell *)cellForRowAtIndexPath:(NSIndexPath *)indexPath
```



```
- (void)deleteRowsAtIndexPaths:(NSArray *)indexPaths
  withRowAnimation:(UITableViewRowAnimation)animation
```

```
- (id)dequeueReusableCellWithIdentifier:(NSString *)identifier
```

```
- (id)dequeueReusableCellWithIdentifier:(NSString *)identifier
  forIndexPath:(NSIndexPath *)indexPath
```

```
- (void)reloadData
```

```
- (void)reloadRowsAtIndexPaths:(NSArray *)indexPaths
  withRowAnimation:(UITableViewRowAnimation)animation
```

```
- (NSArray *)visibleCells
```

示例代码和步骤

1.在ViewController.xib中添加表格视图，如下所示



2. 通过右键单击并选择数据源和委托将委托和数据源设定到"File's Owner（文件的所有者）"。设置数据源如下所示



3.为表格视图创建IBOutlet的并将其命名为myTableView。如以下图片中所示





4. 为拥有数据，添加一个NSMutableArray使其能够在列表格视图中显示

5.ViewController应采用的UITableViewDataSource和UITableViewDelegate协议。ViewController.h代码如下所示

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController<UITableViewDataSource,
UITableViewDelegate>
{

    IBOutlet UITableView *myTableView;
    NSMutableArray *myData;
}

@end
```

6.执行所需的表格视图委托和数据源的方法。更新ViewController.m，如下所示

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    // table view data is being set here
    myData = [[NSMutableArray alloc]initWithObjects:
    @"Data 1 in array",@"Data 2 in array",@"Data 3 in array",
    @"Data 4 in array",@"Data 5 in array",@"Data 5 in array",
    @"Data 6 in array",@"Data 7 in array",@"Data 8 in array",
    @"Data 9 in array", nil];
    // Do any additional setup after loading the view, typically fr
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
```

```
#pragma mark - Table View Data source
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSecti
  (NSInteger)section{
    return [myData count]/2;
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowA
  (NSIndexPath *)indexPath{
    static NSString *cellIdentifier = @"cellID";

    UITableViewCell *cell = [tableView dequeueReusableCellWithIdent
    cellIdentifier];
    if (cell == nil) {
        cell = [[UITableViewCell alloc]initWithStyle:
        UITableViewCellStyleDefault reuseIdentifier:cellIdentifier]
    }
    NSString *stringForCell;
    if (indexPath.section == 0) {
        stringForCell= [myData objectAtIndex:indexPath.row];

    }
    else if (indexPath.section == 1){
        stringForCell= [myData objectAtIndex:indexPath.row+ [myData

    }
    [cell.textLabel setText:stringForCell];
    return cell;
}

// Default is 1 if not implemented
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView{
    return 2;
}

- (NSString *)tableView:(UITableView *)tableView titleForHeaderInSe
  (NSInteger)section{
    NSString *headerTitle;
    if (section==0) {
        headerTitle = @"Section 1 Header";
    }
    else{
        headerTitle = @"Section 2 Header";

    }
    return headerTitle;
}
- (NSString *)tableView:(UITableView *)tableView titleForFooterInSe
  (NSInteger)section{
    NSString *footerTitle;
    if (section==0) {
        footerTitle = @"Section 1 Footer";
    }
```

```
    else{
        footerTitle = @"Section 2 Footer";

    }
    return footerTitle;
}

#pragma mark - TableView delegate

-(void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:
 (NSIndexPath *)indexPath{
    [tableView deselectRowAtIndexPath:indexPath animated:YES];
    UITableViewCell *cell = [tableView cellForRowAtIndexPath:indexP
    NSLog(@"Section:%d Row:%d selected and its data is %@",
    indexPath.section,indexPath.row,cell.textLabel.text);
}

@end
```

7.现在当我们运行应用程序时我们就会得到下面的输出



分割视图的使用

分割视图是 iPad 的特定视图控制器用于管理两个视图控制器，在左侧是一个主控制器，其右侧是一个详细信息视图控制器。 重要的属性

- delegate
- viewControllers

示例代码和步骤

1.创建一个新项目，选择Master Detail Application并单击下一步，输入项目名称，然后选择创建。

2.简单的分割视图控制器与母版中的表视图是默认创建的。

3.在这里我们为我们创建的下列文件。

- AppDelegate.h
- AppDelegate.m
- DetailViewController.h
- DetailViewController.m
- DetailViewController.xib
- MasterViewController.h
- MasterViewController.m
- MasterViewController.xib

4. AppDelegate.h文件如下所示

```
#import <UIKit/UIKit.h>

@interface AppDelegate : UIResponder <UIApplicationDelegate>

@property (strong, nonatomic) UIWindow *window;

@property (strong, nonatomic) UISplitViewController *splitViewContr

@end
```

5.在AppDelegate.m中的didFinishLaunchingWithOptions方法，如下所示

```
  - (BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScr
    bounds]];
     // Override point for customization after application launch.
     MasterViewController *masterViewController = [[MasterViewContro
     alloc] initWithNibName:@"MasterViewController" bundle:nil];
     UINavigationController *masterNavigationController =
     [[UINavigationController alloc] initWithRootViewController:
     masterViewController];

     DetailViewController *detailViewController =
     [[DetailViewController alloc] initWithNibName:@"DetailViewContr
     bundle:nil];
     UINavigationController *detailNavigationController =
     [[UINavigationController alloc] initWithRootViewController:
     detailViewController];

     masterViewController.detailViewController = detailViewControlle

     self.splitViewController = [[UISplitViewController alloc] init]
     self.splitViewController.delegate = detailViewController;
     self.splitViewController.viewControllers =
     @[masterNavigationController, detailNavigationController];
     self.window.rootViewController = self.splitViewController;
     [self.window makeKeyAndVisible];
     return YES;
}
```

6. MasterViewController.h，如下所示

```
#import <UIKit/UIKit.h>

@class DetailViewController;

@interface MasterViewController : UITableViewController

@property (strong, nonatomic) DetailViewController *detailViewCont

@end
```

7. MasterViewController.m，如下所示

```
#import "MasterViewController.h"

#import "DetailViewController.h"

@interface MasterViewController () {
    NSMutableArray *_objects;
}
@end

@implementation MasterViewController

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)
  nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrN
    if (self) {
        self.title = NSLocalizedString(@"Master", @"Master");
        self.clearsSelectionOnViewWillAppear = NO;
        self.contentSizeForViewInPopover = CGSizeMake(320.0, 600.0
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    self.navigationItem.leftBarButtonItem = self.editButtonItem;

    UIBarButtonItem *addButton = [[UIBarButtonItem alloc]
    initWithBarButtonSystemItem: UIBarButtonSystemItemAdd
    target:self action:@selector(insertNewObject:)];
    self.navigationItem.rightBarButtonItem = addButton;
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
```

```
}

- (void)insertNewObject:(id)sender
{
    if (!_objects) {
        _objects = [[NSMutableArray alloc] init];
    }
    [_objects insertObject:[NSDate date] atIndex:0];
    NSIndexPath *indexPath = [NSIndexPath indexPathForRow:0 inSecti
    [self.tableView insertRowsAtIndexPaths:@[indexPath] withRowAnin
    UITableViewRowAnimationAutomatic];
}

#pragma mark - Table View

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSecti
  (NSInteger)section
{
    return _objects.count;
}

// Customize the appearance of table view cells.
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowA
  (NSIndexPath *)indexPath
{
    static NSString *CellIdentifier = @"Cell";

    UITableViewCell *cell = [tableView dequeueReusableCellWithIdent
    CellIdentifier];
    if (cell == nil) {
        cell = [[UITableViewCell alloc] initWithStyle:
        UITableViewCellStyleDefault reuseIdentifier:CellIdentifier]
    }

    NSDate *object = _objects[indexPath.row];
    cell.textLabel.text = [object description];
    return cell;
}

- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:
  (NSIndexPath *)indexPath
{
    // Return NO if you do not want the specified item to be editab
    return YES;
}

- (void)tableView:(UITableView *)tableView commitEditingStyle:
  (UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:
```

```
  (NSIndexPath *)indexPath
{
    if (editingStyle == UITableViewCellEditingStyleDelete) {
        [_objects removeObjectAtIndex:indexPath.row];
        [tableView deleteRowsAtIndexPaths:@[indexPath] withRowAnima
        UITableViewRowAnimationFade];
    } else if (editingStyle == UITableViewCellEditingStyleInsert) {
        // Create a new instance of the appropriate class, insert i
        //the array, and add a new row to the table view.
    }
}

/*
// Override to support rearranging the table view.
- (void)tableView:(UITableView *)tableView moveRowAtIndexPath:
  (NSIndexPath *) fromIndexPath toIndexPath:(NSIndexPath *)toIndexP
{
}
*/

/*
// Override to support conditional rearranging of the table view.
- (BOOL)tableView:(UITableView *)tableView canMoveRowAtIndexPath:
  (NSIndexPath *)indexPath
{
    // Return NO if you do not want the item to be re-orderable.
    return YES;
}
*/

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:
  (NSIndexPath *)indexPath
{
    NSDate *object = _objects[indexPath.row];
    self.detailViewController.detailItem = object;
    NSDateFormatter *formatter = [[NSDateFormatter alloc] init];
    [formatter setDateFormat: @"yyyy-MM-dd HH:mm:ss zzz"];
    NSString *stringFromDate = [formatter stringFromDate:object];
    self.detailViewController.detailDescriptionLabel.text = stringF
}

@end
```

8. DetailViewController.h， 如下所示

```
#import <UIKit/UIKit.h>

@interface DetailViewController : UIViewController
<UISplitViewControllerDelegate>

@property (strong, nonatomic) id detailItem;

@property (weak, nonatomic) IBOutlet UILabel *detailDescriptionLabe
@end
```

9. DetailViewController.m， 如下所示

```
#import "DetailViewController.h"

@interface DetailViewController ()
@property (strong, nonatomic) UIPopoverController *masterPopoverCor
- (void)configureView;
@end

@implementation DetailViewController

#pragma mark - Managing the detail item

- (void)setDetailItem:(id)newDetailItem
{
    if (_detailItem != newDetailItem) {
        _detailItem = newDetailItem;

        // Update the view.
        [self configureView];
    }

    if (self.masterPopoverController != nil) {
        [self.masterPopoverController dismissPopoverAnimated:YES];
    }
}

- (void)configureView
{
    // Update the user interface for the detail item.

    if (self.detailItem) {
        self.detailDescriptionLabel.text = [self.detailItem descrip
    }
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self configureView];
```

```
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:
  (NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNi
    if (self) {
        self.title = NSLocalizedString(@"Detail", @"Detail");
    }
    return self;
}

#pragma mark - Split view

- (void)splitViewController:(UISplitViewController *)splitControlle
  willHideViewController:(UIViewController *)viewController withBar
  (UIBarButtonItem *)barButtonItem forPopoverController:
  (UIPopoverController *)popoverController
{
    barButtonItem.title = NSLocalizedString(@"Master", @"Master");
    [self.navigationItem setLeftBarButtonItem:barButtonItem animate
    self.masterPopoverController = popoverController;
}

- (void)splitViewController:(UISplitViewController *)splitControlle
  willShowViewController:(UIViewController *)viewController
  invalidatingBarButtonItem:(UIBarButtonItem *)barButtonItem
{
    // Called when the view is shown again in the split view,
    //invalidating the button and popover controller.
    [self.navigationItem setLeftBarButtonItem:nil animated:YES];
    self.masterPopoverController = nil;
}

@end
```

10.现在当我们运行应用程序时，在横向模式下我们会得到下面的输出

11. 当我们切换到纵向模式，我们会获得下面的输出:

IOS文本视图的使用

文本视图用于显示多行滚动的文本。

重要属性

- dataDetectorTypes
- delegate
- editable
- inputAccessoryView
- inputView
- text
- textAlignment
- textColor

重要的委托方法

```
-(void)textViewDidBeginEditing:(UITextView *)textView
```

```
-(void)textViewDidEndEditing:(UITextView *)textView
```

```
-(void)textViewDidChange:(UITextView *)textView
```

```
-(BOOL)textViewShouldEndEditing:(UITextView *)textView
```

添加自定义方法 **addTextView**

```
-(void)addTextView{
    myTextView = [[UITextView alloc]initWithFrame:
    CGRectMake(10, 50, 300, 200)];
    [myTextView setText:@"Lorem ipsum dolor sit er elit lamet, cons
    cillium adipisicing pecu, sed do eiusmod tempor incididunt ut l
    dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exer
    ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis a
    dolor in reprehenderit in voluptate velit esse cillum dolore eu
    nulla pariatur. Excepteur sint occaecat cupidatat non proident,
    culpa qui officia deserunt mollit anim id est laborum. Nam libe
    conscient to factor tum poen legum odioque civiuda.
    Lorem ipsum dolor sit er elit lamet, consectetaur cillium adipi
    pecu, sed do eiusmod tempor incididunt ut labore et dolore magn
    Ut enim ad minim veniam, quis nostrud exercitation ullamco labo
    aliquip ex ea commodo consequat. Duis aute irure dolor in repre
    in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
    Excepteur sint occaecat cupidatat non proident, sunt in culpa
    qui officia deserunt mollit anim id est laborum. Nam liber te c
    to factor tum poen legum odioque civiuda."];
    myTextView.delegate = self;
    [self.view addSubview:myTextView];

}
```

在 ViewController.m 中执行 textView 委托

```
#pragma mark - Text View delegates

-(BOOL)textView:(UITextView *)textView shouldChangeTextInRange:
 (NSRange)range replacementText:(NSString *)text{
    if ([text isEqualToString:@"\n"]) {
        [textView resignFirstResponder];
    }
    return YES;
}
-(void)textViewDidBeginEditing:(UITextView *)textView{
    NSLog(@"Did begin editing");
}
-(void)textViewDidChange:(UITextView *)textView{
    NSLog(@"Did Change");

}
-(void)textViewDidEndEditing:(UITextView *)textView{
    NSLog(@"Did End editing");

}
-(BOOL)textViewShouldEndEditing:(UITextView *)textView{
    [textView resignFirstResponder];
    return YES;
}
```

修改 ViewController.m 中的 viewDidLoad方法，如下所示

```
(void)viewDidLoad
{
   [super viewDidLoad];
   [self addTextView];
}
```

结果输出

现在当我们运行该应用程序我们会看到下面的输出

IOS视图切换的使用

视图切换通过一系列动画效果实现,包括折叠切换、爆炸切换、卡片式切换等等。

修改 ViewController.xib，如下所示

在 xib 中创建按钮的操作

修改 ViewController.h

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController
{
    UIView *view1;
    UIView *view2;
}

-(IBAction)flipFromLeft:(id)sender;
-(IBAction)flipFromRight:(id)sender;
-(IBAction)flipFromTop:(id)sender;
-(IBAction)flipFromBottom:(id)sender;
-(IBAction)curlUp:(id)sender;
-(IBAction)curlDown:(id)sender;
-(IBAction)dissolve:(id)sender;
-(IBAction)noTransition:(id)sender;

@end
```

在 ViewController 类中声明两个视图的实例。ViewController.h文件代码如下：

修改 ViewController.m

我们将添加自定义方法setUpView来初始化视图。

我们还将创建了另一种方法doTransitionWithType: 实现view1切换到view2，反之亦然。

后我们将执行之前创建的操作的方法即调用 doTransitionWithType: 方法与切换类型。ViewController.m代码如下:

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self setUpView];
    // Do any additional setup after loading the view, typically fr
}
```

```
-(void)setUpView{
    view1 = [[UIView alloc]initWithFrame:self.view.frame];
    view1.backgroundColor = [UIColor lightTextColor];
    view2 = [[UIView alloc]initWithFrame:self.view.frame];
    view2.backgroundColor = [UIColor orangeColor];
    [self.view addSubview:view1];
    [self.view sendSubviewToBack:view1];

}

-(void)doTransitionWithType:(UIViewAnimationTransition)animationTra
    if ([[self.view subviews] containsObject:view2 ]) {
        [UIView transitionFromView:view2
                            toView:view1
                          duration:2
                           options:animationTransitionType
                        completion:^(BOOL finished){
                            [view2 removeFromSuperview];
                        }];
        [self.view addSubview:view1];
        [self.view sendSubviewToBack:view1];
    }
    else{

        [UIView transitionFromView:view1
                            toView:view2
                          duration:2
                           options:animationTransitionType
                        completion:^(BOOL finished){
                            [view1 removeFromSuperview];
                        }];
        [self.view addSubview:view2];
        [self.view sendSubviewToBack:view2];

    }
}

-(IBAction)flipFromLeft:(id)sender
{
    [self doTransitionWithType:UIViewAnimationOptionTransitionFlipF

}
-(IBAction)flipFromRight:(id)sender{
    [self doTransitionWithType:UIViewAnimationOptionTransitionFlipF

}
-(IBAction)flipFromTop:(id)sender{
    [self doTransitionWithType:UIViewAnimationOptionTransitionFlipF

}
-(IBAction)flipFromBottom:(id)sender{
    [self doTransitionWithType:UIViewAnimationOptionTransitionFlipF
```

```
}
-(IBAction)curlUp:(id)sender{
    [self doTransitionWithType:UIViewAnimationOptionTransitionCurlU

}
-(IBAction)curlDown:(id)sender{
    [self doTransitionWithType:UIViewAnimationOptionTransitionCurlD

}
-(IBAction)dissolve:(id)sender{
    [self doTransitionWithType:UIViewAnimationOptionTransitionCross

}
-(IBAction)noTransition:(id)sender{
    [self doTransitionWithType:UIViewAnimationOptionTransitionNone]

}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```

输出

现在当我们运行该应用程序我们会看到下面的输出

您可以选择不同的按钮，看切换是如何工作。选择蜷缩切换将效果如下所示

选择器的使用

选择器是一个可滚动视图，用于选取列表项中的值。

重要的属性

- delegate
- dataSource

重要的方法

```
- (void)reloadAllComponents
```

```
- (void)reloadComponent:(NSInteger)component
```

```
- (NSInteger)selectedRowInComponent:(NSInteger)component
```

```
- (void)selectRow:(NSInteger)row inComponent:(NSInteger)component
  animated:(BOOL)animated
```

修改 ViewController.h

我们将添加一个文本字段、选择器视图和一个数组。

我们将采用UITextFieldDelegate、UIPickerViewDataSource、UIPickerViewDelegate的协议。ViewController.h文件代码如下所示：

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController
<UITextFieldDelegate,UIPickerViewDataSource,UIPickerViewDelegate>
{

    UITextField *myTextField;
    UIPickerView *myPickerView;
    NSArray *pickerArray;
}
@end
```

添加自定义方法 addPickerView

```
-(void)addPickerView{
    pickerArray = [[NSArray alloc]initWithObjects:@"Chess",
    @"Cricket",@"Football",@"Tennis",@"Volleyball", nil];
    myTextField = [[UITextField alloc]initWithFrame:
    CGRectMake(10, 100, 300, 30)];
    myTextField.borderStyle = UITextBorderStyleRoundedRect;
    myTextField.textAlignment = UITextAlignmentCenter;
    myTextField.delegate = self;
    [self.view addSubview:myTextField];
    [myTextField setPlaceholder:@"Pick a Sport"];
    myPickerView = [[UIPickerView alloc]init];
    myPickerView.dataSource = self;
    myPickerView.delegate = self;
    myPickerView.showsSelectionIndicator = YES;
    UIBarButtonItem *doneButton = [[UIBarButtonItem alloc]
    initWithTitle:@"Done" style:UIBarButtonItemStyleDone
    target:self action:@selector(done:)];
    UIToolbar *toolBar = [[UIToolbar alloc]initWithFrame:
    CGRectMake(0, self.view.frame.size.height-
    myDatePicker.frame.size.height-50, 320, 50)];
    [toolBar setBarStyle:UIBarStyleBlackOpaque];
    NSArray *toolbarItems = [NSArray arrayWithObjects:
    doneButton, nil];
    [toolBar setItems:toolbarItems];
    myTextField.inputView = myPickerView;
    myTextField.inputAccessoryView = toolBar;

}
```

执行委托，如下所示:

```
#pragma mark - Text field delegates

-(void)textFieldDidBeginEditing:(UITextField *)textField{
    if ([textField.text isEqualToString:@""]) {
        [self dateChanged:nil];
    }
}
#pragma mark - Picker View Data source
-(NSInteger)numberOfComponentsInPickerView:(UIPickerView *)pickerVi
    return 1;
}
-(NSInteger)pickerView:(UIPickerView *)pickerView
 numberOfRowsInComponent:(NSInteger)component{
    return [pickerArray count];
}

#pragma mark- Picker View Delegate

-(void)pickerView:(UIPickerView *)pickerView didSelectRow:
 (NSInteger)row inComponent:(NSInteger)component{
    [myTextField setText:[pickerArray objectAtIndex:row]];
}
- (NSString *)pickerView:(UIPickerView *)pickerView titleForRow:
  (NSInteger)row forComponent:(NSInteger)component{
    return [pickerArray objectAtIndex:row];
}
```

在ViewController.m修改viewDidLoad，如下所示:

```
(void)viewDidLoad
{
   [super viewDidLoad];
   [self addPickerView];
}
```

输出

现在当我们运行该应用程序我们会看到下面的输出:

文本选择器视图如下所示，我们可以选取我们需要的值：

IOS开关的使用

开关用于打开和关闭状态之间的切换。

重要的属性

- onImage
- offImage
- on

重要的方法

```
- (void)setOn:(BOOL)on animated:(BOOL)animated
```

添加自定义方法 **addSwitch** 和开关

```
-(IBAction)switched:(id)sender{
    NSLog(@"Switch current state %@", mySwitch.on ? @"On" : @"Off")
}
-(void)addSwitch{
    mySwitch = [[UISwitch alloc] init];
    [self.view addSubview:mySwitch];
    mySwitch.center = CGPointMake(150, 200);
    [mySwitch addTarget:self action:@selector(switched:)
    forControlEvents:UIControlEventValueChanged];
}
```

在 **ViewController.m** 中修改 **viewDidLoad**，如下所示

```
(void)viewDidLoad
{
   [super viewDidLoad];
   [self addSwitch];
}
```

输出

现在当我们运行该应用程序我们会看到下面的输出

向右滑动开关输出如下所示

IOS滑块的使用

滑块用于从某个范围的值里选择的一个值。

重要的属性

- continuous
- maximumValue
- minimumValue
- value

重要的方法

```
- (void)setValue:(float)value animated:(BOOL)animated
```

添加自定义方法 **addSlider** 和 **sliderChanged**

```
-(IBAction)sliderChanged:(id)sender{
    NSLog(@"SliderValue %f",mySlider.value);
}
-(void)addSlider{
    mySlider = [[UISlider alloc] initWithFrame:CGRectMake(50, 200,
    [self.view addSubview:mySlider];
    mySlider.minimumValue = 10.0;
    mySlider.maximumValue = 99.0;
    mySlider.continuous = NO;
    [mySlider addTarget:self action:@selector(sliderChanged:)
    forControlEvents:UIControlEventValueChanged];
}
```

在 **ViewController.m** 中修改 **viewDidLoad**，如下所示

```
(void)viewDidLoad
{
   [super viewDidLoad];
   [self addSlider];
}
```

输出

现在当我们运行该应用程序我们会看到下面的输出

当拖动滑块效果如下：



IOS警告对话框的使用

警告对话框用来给用户提供重要信息。

仅在警告对话框视图中选择选项后，才能着手进一步使用应用程序。

重要的属性

- alertViewStyle
- cancelButtonIndex
- delegate
- message
- numberOfButtons
- title

重要的方法

```
- (NSInteger)addButtonWithTitle:(NSString *)title
```

```
- (NSString *)buttonTitleAtIndex:(NSInteger)buttonIndex
```

```
- (void)dismissWithClickedButtonIndex:
  (NSInteger)buttonIndex animated:(BOOL)animated
```

```
- (id)initWithTitle:(NSString *)title message:
  (NSString *)message delegate:(id)delegate
  cancelButtonTitle:(NSString *)cancelButtonTitle
  otherButtonTitles:(NSString*)otherButtonTitles, ...
```

```
- (void)show
```

更新 **ViewController.h**，如下所示

让类符合警告对话框视图的委托协议，如下所示，在ViewController.h中添加 <UIAlertViewDelegate>

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController<UIAlertViewDelegate>{

}
@end
```

添加自定义方法 addAlertView

```
-(void)addAlertView{
    UIAlertView *alertView = [[UIAlertView alloc]initWithTitle:
    @"Title" message:@"This is a test alert" delegate:self
    cancelButtonTitle:@"Cancel" otherButtonTitles:@"Ok", nil];
    [alertView show];
}
```

执行警告对话框视图的委托方法

```
#pragma mark - Alert view delegate
- (void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:
  (NSInteger)buttonIndex{
    switch (buttonIndex) {
        case 0:
            NSLog(@"Cancel button clicked");
            break;
        case 1:
            NSLog(@"OK button clicked");
            break;

        default:
            break;
    }
}
```

在 ViewController.m 中修改 viewDidLoad，如下所示

```
(void)viewDidLoad
{
   [super viewDidLoad];
   [self addAlertView];
}
```

输出

现在当我们运行该应用程序我们会看到下面的输出：



IOS图标的使用

IOS图标是用于应用程序相关的操作。

IOS 中的不同图标

- AppIcon
- App Store 的应用程序图标
- 搜索结果和设置的小图标
- 工具栏和导航栏图标
- 选项卡栏图标

AppIcon

AppIcon 是出现在设备SpringBoard （默认屏幕上的所有的应用程序） 的应用程序的图标。

App Store 的应用程序图标

它是512 x 512 或 1024 x 1024(推荐大小)的高分辨率的应用程序图标。

搜索结果和设置的小图标

在搜索列表的应用程序中使用这个小图标。

它还用于与相关的应用程序的功能是启用和禁用的设置屏幕上。如：启用定位服务。

工具栏和导航栏图标

工具栏和导航栏中使用特制的标准图标的列表。它包括的份额，像图标相机，撰写等等。

选项卡栏图标

选项卡栏中使用一系列特制的标准图标列表。它包括的图标有书签、 联系人、 下载等。

有的不同的 iOS 设备的每个图标大小的都不一样。你可以查看更多关于苹果文件中图标的准则：ios人机交互界面指南。

IOS加速度传感器(accelerometer)

简介

加速度传感器是根据x、y和z三个方向来检测在设备位置的改变。

通过加速度传感器可以知道当前设备相对于地面的位置。

以下实例代码需要在真实设备上运行，在模拟器上是无法工作的。

实例步骤

1、创建一个简单的视图应用程序

2、在ViewController.xib中添加三个标签，并创建一个ibOutlets分别为：xlable、ylabel和zlabel

3、如下所示，更新ViewController.h

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController<UIAccelerometerDelegat
{
    IBOutlet UILabel *xlabel;
    IBOutlet UILabel *ylabel;
    IBOutlet UILabel *zlabel;

}
@end
```

4、如下所示，更新ViewController.m

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [[UIAccelerometer sharedAccelerometer]setDelegate:self];
    //Do any additional setup after loading the view,typically fror
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
- (void)accelerometer:(UIAccelerometer *)accelerometer didAccelerat
  (UIAcceleration *)acceleration{
    [xlabel setText:[NSString stringWithFormat:@"%f",acceleration.x
    [ylabel setText:[NSString stringWithFormat:@"%f",acceleration.y
    [zlabel setText:[NSString stringWithFormat:@"%f",acceleration.z
}

@end
```

输出

当我们在iPhone设备中运行该应用程序，得到的输出结果如下所示。


No SIM
9:01 AM
87%
-0.036697
-0.886536
-0.421844

IOS通用应用程序

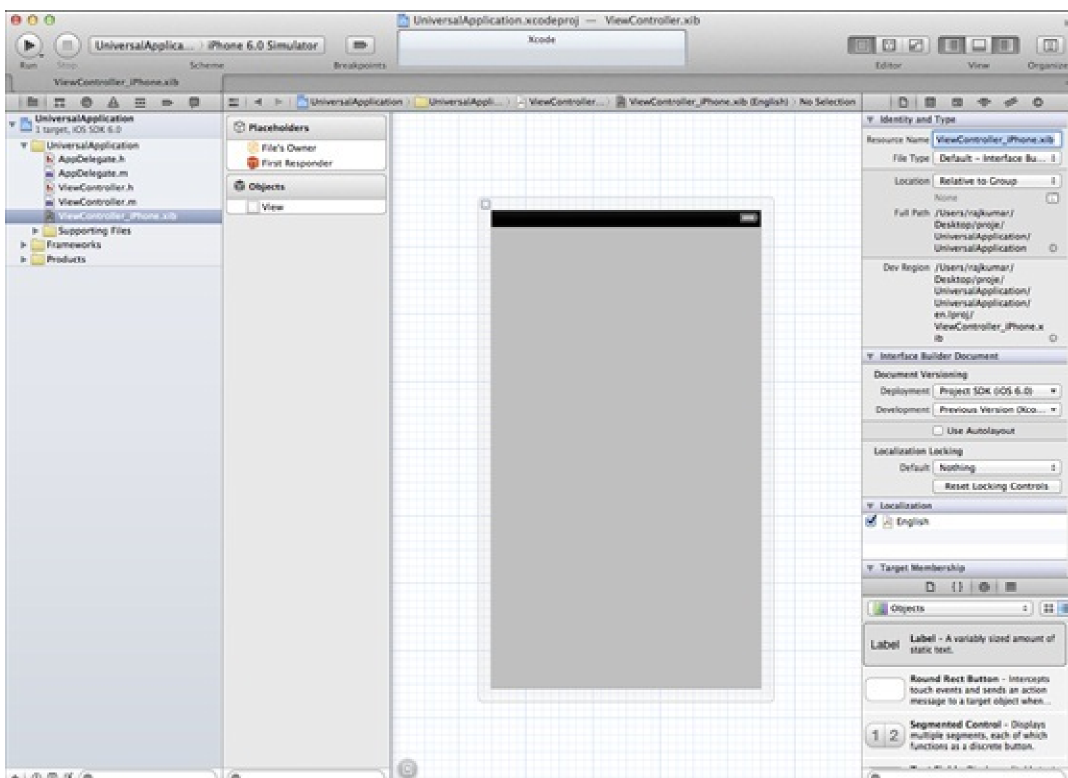
简介

通用的应用程序是为iPhone和iPad在一个单一的二进制文件中设计的应用程序。这有助于代码重用，并能够帮助更快进行更新。

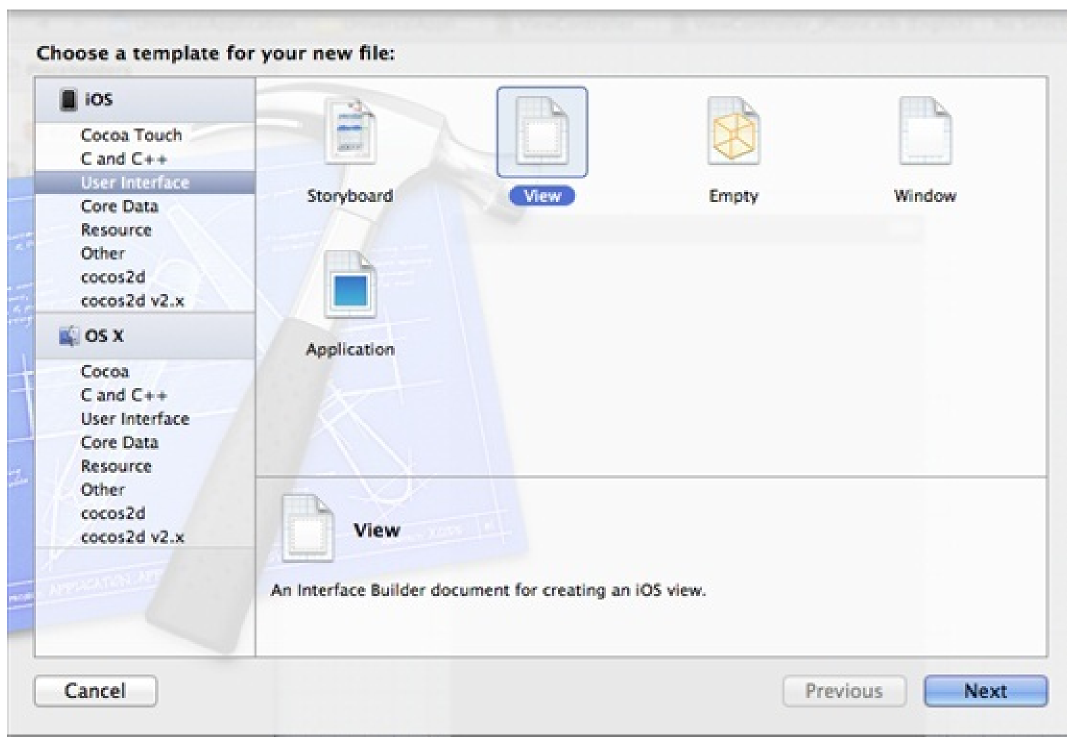
实例步骤

1、创建一个简单的View based application（视图应用程序）

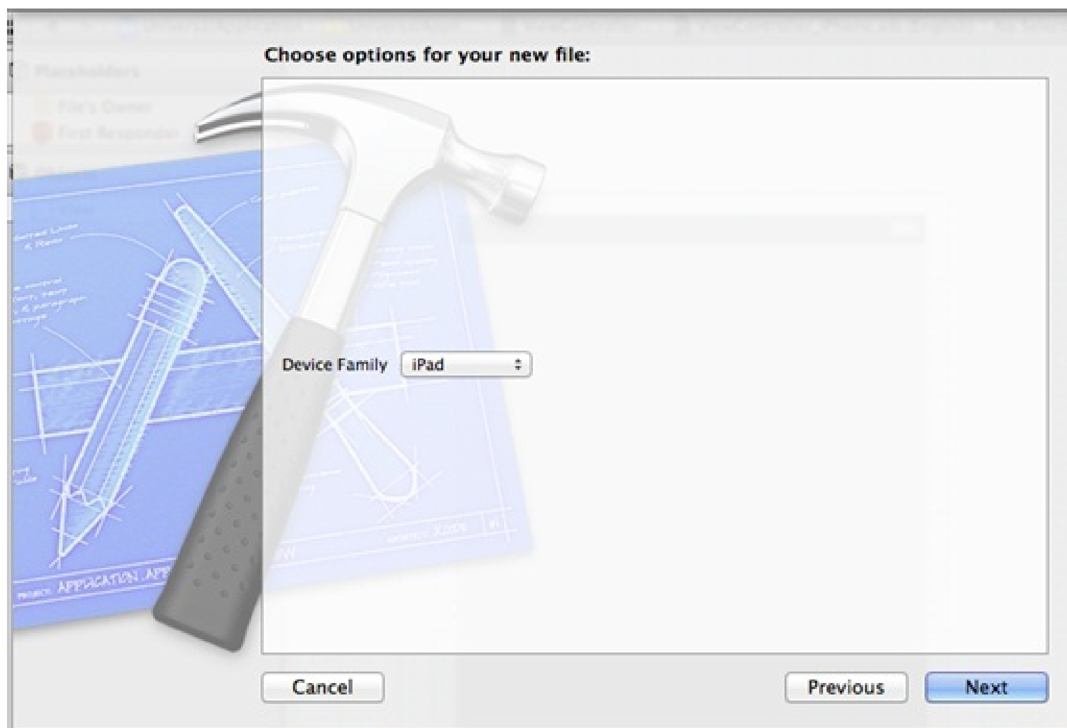
2、在文件查看器的右边，将文件ViewController.xib的文件名称更改为ViewController_iPhone.xib，如下所示



3、选择"File -> New -> File... "，然后选择User Interface，再选择View，单击下一步



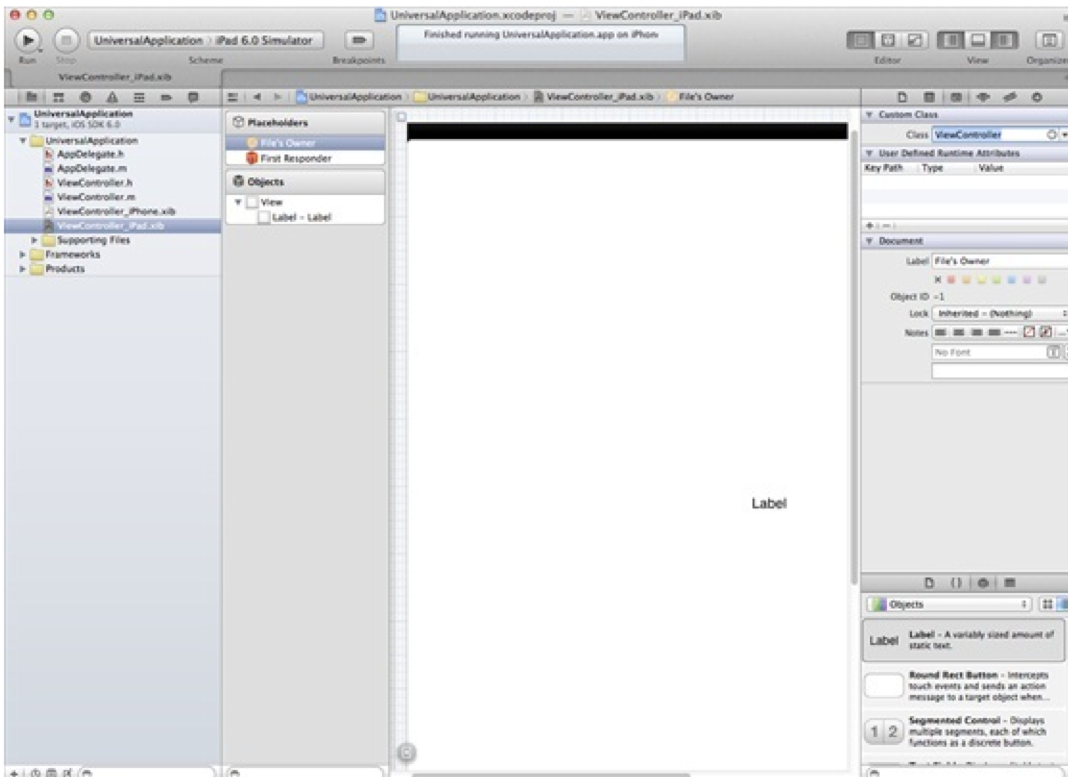
4、选择iPad作为设备，单击下一步:



5、将该文件另存为ViewController_iPad.xib，然后选择创建

6、在ViewController_iPhone.xib和ViewController_iPad.xibd的屏幕中心添加标签

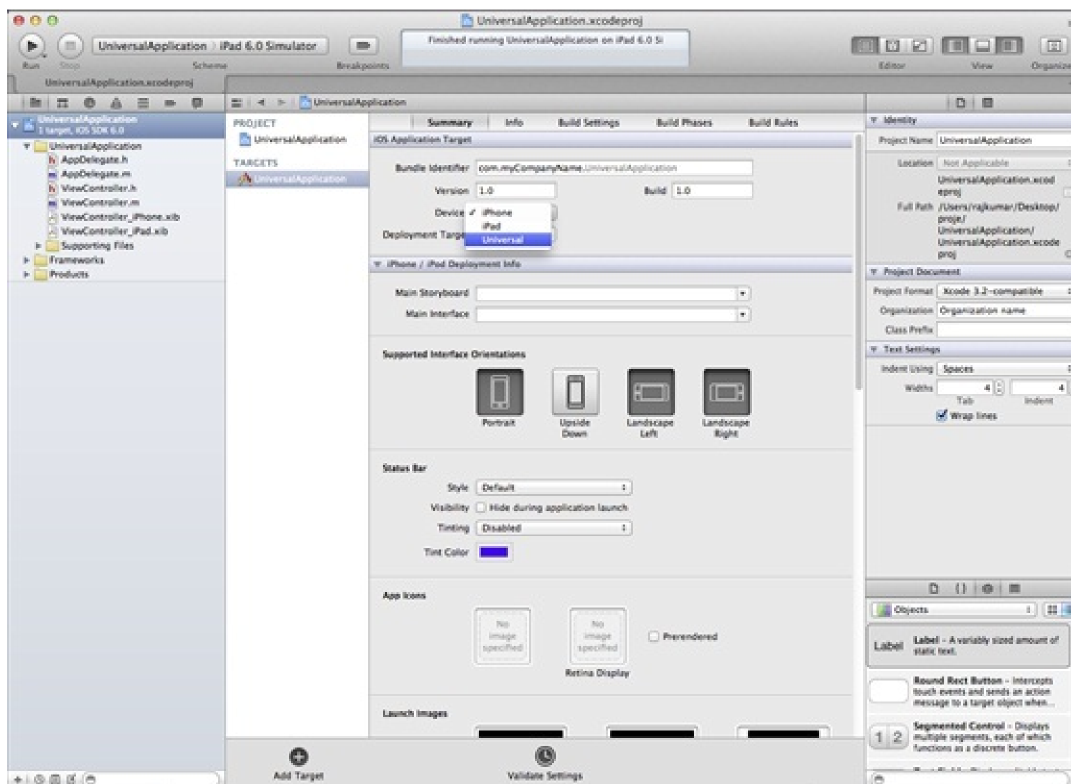
7、在ViewController_iPhone.xib中选择identity inspector，设置custom class为ViewController



8、更新AppDelegate.m中的 application:DidFinishLaunching:withOptions方法

```
- (BOOL)application:(UIApplication *)application
  didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
   self.window = [[UIWindow alloc] initWithFrame:[[UIScreen
   mainScreen] bounds]];
   // Override point for customization after application launch.
   if (UI_USER_INTERFACE_IDIOM() == UIUserInterfaceIdiomPhone) {
        self.viewController = [[ViewController alloc]
        initWithNibName:@"ViewController_iPhone" bundle:nil];
   }
   else{
        self.viewController = [[ViewController alloc] initWithNibNa
        @"ViewController_iPad" bundle:nil];
   }
   self.window.rootViewController = self.viewController;
   [self.window makeKeyAndVisible];
   return YES;
}
```

9、在项目摘要中更新设备中为universal，如下所示：

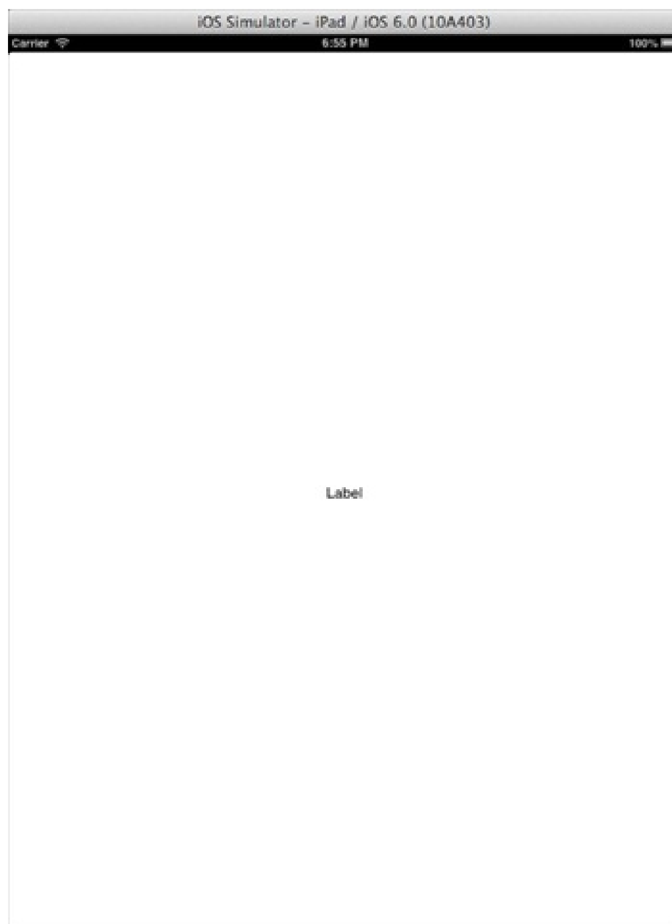


输出

运行该应用程序，我们会看到下面的输出



在iPad模拟器中运行应用程序,我们会得到下面的输出:



IOS相机管理

相机简介

相机是移动设备的共同特点之一，我们能够使用相机拍摄图片，并在应用程序里调用它，而且相机的使用很简单。

实例步骤

1、创建一个简单的View based application

2、在ViewController.xib中添加一个button （按钮），并为该按钮创建IBAction

3、添加一个 image view （图像视图），并创建一个名为imageView的IBOutlet

4、ViewController.h文件代码如下所示：

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController<UIImagePickerControlle
{
   UIImagePickerController *imagePicker;
   IBOutlet UIImageView *imageView;
}
- (IBAction)showCamera:(id)sender;

@end
```

5、修改ViewController.m,如下所示：

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (IBAction)showCamera:(id)sender {
    imagePicker.allowsEditing = YES;
    if ([UIImagePickerController isSourceTypeAvailable:
        UIImagePickerControllerSourceTypeCamera])
    {
        imagePicker.sourceType = UIImagePickerControllerSourceType(
    }
    else{
        imagePicker.sourceType =
        UIImagePickerControllerSourceTypePhotoLibrary;
    }
    [self presentModalViewController:imagePicker animated:YES];

}
-(void)imagePickerController:(UIImagePickerController *)picker
  didFinishPickingMediaWithInfo:(NSDictionary *)info{
    UIImage *image = [info objectForKey:UIImagePickerControllerEdit
    if (image == nil) {
        image = [info objectForKey:UIImagePickerControllerOriginalI
    }
    imageView.image = image;

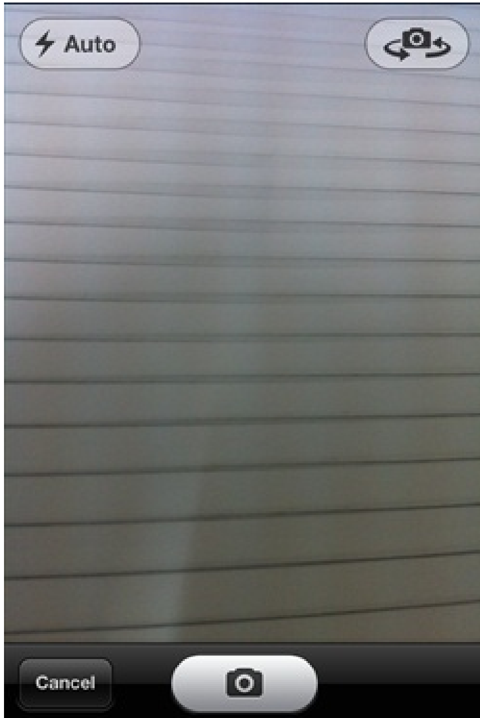
}

-(void)imagePickerControllerDidCancel:(UIImagePickerController *)pi
    [self dismissModalViewControllerAnimated:YES];
}

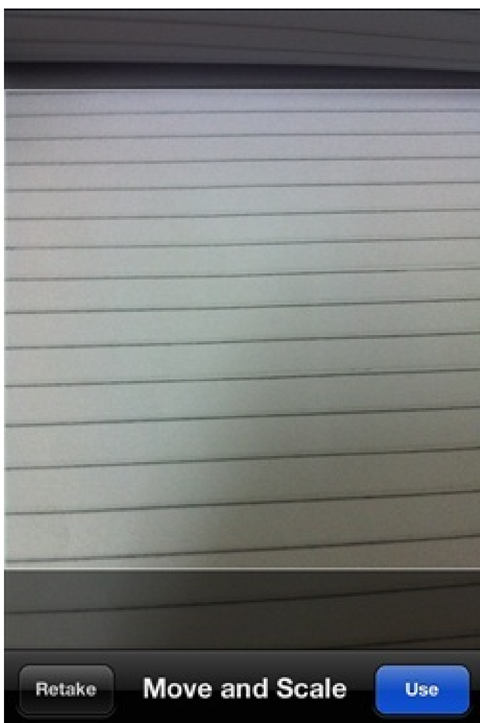
@end
```

输出

运行该应用程序并单击显示相机按钮时，我们就会获得下面的输出



只要拍照之后，就可以通过移动和缩放对图片进行编辑，如下所示。



IOS定位操作

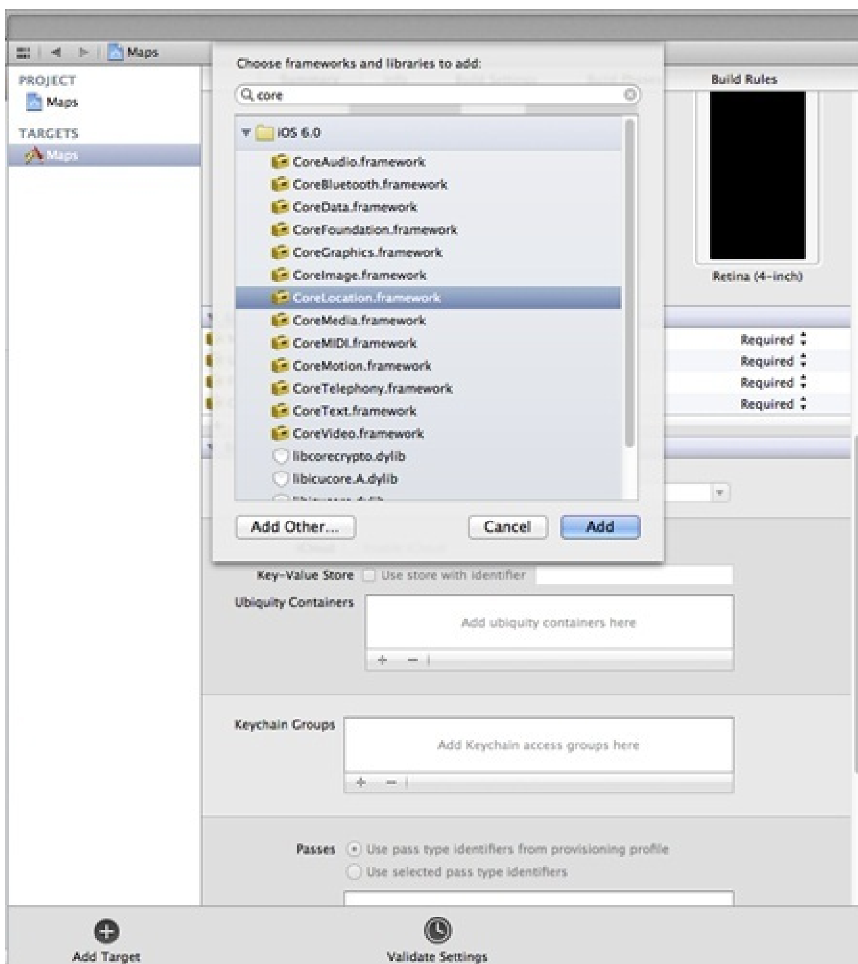
简介

在IOS中通过CoreLocation定位，可以获取到用户当前位置，同时能得到装置移动信息。

实例步骤

1、创建一个简单的View based application（视图应用程序）。

2、择项目文件，然后选择目标，然后添加CoreLocation.framework,如下所示



3、在ViewController.xib中添加两个标签，创建ibOutlet名为latitudeLabel和longtitudeLabel的标签

4、现在通过选择" File-> New -> File... -> "选择Objective C class 并单击下一步

5、把"sub class of"作为NSObject，将类命名为LocationHandler

6、选择创建

7、更新LocationHandler.h，如下所示

```
#import <Foundation/Foundation.h>
#import <CoreLocation/CoreLocation.h>

@protocol LocationHandlerDelegate <NSObject>

@required
-(void) didUpdateToLocation:(CLLocation*)newLocation
 fromLocation:(CLLocation*)oldLocation;
@end

@interface LocationHandler : NSObject<CLLocationManagerDelegate>
{
    CLLocationManager *locationManager;
}
@property(nonatomic,strong) id<LocationHandlerDelegate> delegate;

+(id)getSharedInstance;
-(void)startUpdating;
-(void) stopUpdating;

@end
```

8、更新LocationHandler.m,如下所示

```
#import "LocationHandler.h"
static LocationHandler *DefaultManager = nil;

@interface LocationHandler()

-(void)initiate;

@end

@implementation LocationHandler

+(id)getSharedInstance{
    if (!DefaultManager) {
        DefaultManager = [[self allocWithZone:NULL]init];
        [DefaultManager initiate];
    }
    return DefaultManager;
}
-(void)initiate{
    locationManager = [[CLLocationManager alloc]init];
    locationManager.delegate = self;
}

-(void)startUpdating{
    [locationManager startUpdatingLocation];
}

-(void) stopUpdating{
    [locationManager stopUpdatingLocation];
}
-(void)locationManager:(CLLocationManager *)manager didUpdateToLoca
 (CLLocation *)newLocation fromLocation:(CLLocation *)oldLocation{
    if ([self.delegate respondsToSelector:@selector
    (didUpdateToLocation:fromLocation:)])
    {
        [self.delegate didUpdateToLocation:oldLocation
        fromLocation:newLocation];

    }
}

@end
```

9、更新ViewController.h,如下所示

```
#import <UIKit/UIKit.h>
#import "LocationHandler.h"
@interface ViewController : UIViewController<LocationHandlerDelegat
{
    IBOutlet UILabel *latitudeLabel;
    IBOutlet UILabel *longitudeLabel;
}
@end
```

10、更新ViewController.m,如下所示

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [[LocationHandler getSharedInstance]setDelegate:self];
    [[LocationHandler getSharedInstance]startUpdating];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

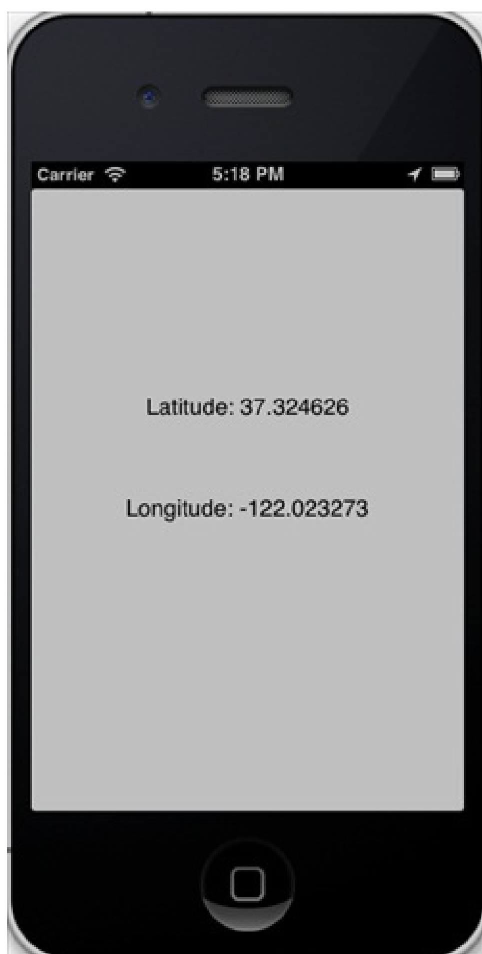
-(void)didUpdateToLocation:(CLLocation *)newLocation
 fromLocation:(CLLocation *)oldLocation{
    [latitudeLabel setText:[NSString stringWithFormat:
    @"Latitude: %f",newLocation.coordinate.latitude]];
    [longitudeLabel setText:[NSString stringWithFormat:
    @"Longitude: %f",newLocation.coordinate.longitude]];

}

@end
```

输出

当我们运行该应用程序，会得到如下的输出:


Carrier
5:18 PM
Latitude: 37.324626
Longitude: -122.023273

IOS SQLite数据库

简介

在IOS中使用Sqlite来处理数据。如果你已经了解了SQL，那你可以很容易的掌握SQLite数据库的操作。

实例步骤

1、创建一个简单的View based application

2、选择项目文件，然后选择目标，添加libsqlite3.dylib库到选择框架

3、通过选择" File-> New -> File... -> "选择 Objective C class 创建新文件，单击下一步

4、"sub class of"为NSObject"，类命名为DBManager

5、选择创建

6、更新DBManager.h,如下所示

```
#import <Foundation/Foundation.h>
#import <sqlite3.h>

@interface DBManager : NSObject
{
    NSString *databasePath;
}

+(DBManager*)getSharedInstance;
-(BOOL)createDB;
-(BOOL) saveData:(NSString*)registerNumber name:(NSString*)name
  department:(NSString*)department year:(NSString*)year;
-(NSArray*) findByRegisterNumber:(NSString*)registerNumber;

@end
```

7、更新DBManager.m,如下所示

```
#import "DBManager.h"
static DBManager *sharedInstance = nil;
static sqlite3 *database = nil;
static sqlite3_stmt *statement = nil;

@implementation DBManager
```

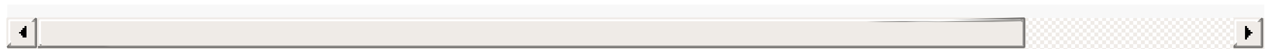
```
+(DBManager*)getSharedInstance{
    if (!sharedInstance) {
        sharedInstance = [[super allocWithZone:NULL]init];
        [sharedInstance createDB];
    }
    return sharedInstance;
}

-(BOOL)createDB{
    NSString *docsDir;
    NSArray *dirPaths;
    // Get the documents directory
    dirPaths = NSSearchPathForDirectoriesInDomains
    (NSDocumentDirectory, NSUserDomainMask, YES);
    docsDir = dirPaths[0];
    // Build the path to the database file
    databasePath = [[NSString alloc] initWithString:
    [docsDir stringByAppendingPathComponent: @"student.db"]];
    BOOL isSuccess = YES;
    NSFileManager *filemgr = [NSFileManager defaultManager];
    if ([filemgr fileExistsAtPath: databasePath ] == NO)
    {
        const char *dbpath = [databasePath UTF8String];
        if (sqlite3_open(dbpath, &database) == SQLITE_OK)
        {
            char *errMsg;
            const char *sql_stmt =
            "create table if not exists studentsDetail (regno inte
            primary key, name text, department text, year text)";
            if (sqlite3_exec(database, sql_stmt, NULL, NULL, &errM
               != SQLITE_OK)
            {
                isSuccess = NO;
                NSLog(@"Failed to create table");
            }
            sqlite3_close(database);
            return  isSuccess;
        }
        else {
            isSuccess = NO;
            NSLog(@"Failed to open/create database");
        }
    }
    return isSuccess;
}

- (BOOL) saveData:(NSString*)registerNumber name:(NSString*)name
  department:(NSString*)department year:(NSString*)year;
{
    const char *dbpath = [databasePath UTF8String];
    if (sqlite3_open(dbpath, &database) == SQLITE_OK)
    {
```

```
        NSString *insertSQL = [NSString stringWithFormat:@"insert i
        studentsDetail (regno,name, department, year) values
        (\"%d\",\"%@\", \"%@\", \"%@\")",[registerNumber integerVal
        name, department, year];
        const char *insert_stmt = [insertSQL UTF8String];
        sqlite3_prepare_v2(database, insert_stmt,-1, &statement, NU
        if (sqlite3_step(statement) == SQLITE_DONE)
        {
            return YES;
        }
        else {
            return NO;
        }
        sqlite3_reset(statement);
    }
    return NO;
}

- (NSArray*) findByRegisterNumber:(NSString*)registerNumber
{
    const char *dbpath = [databasePath UTF8String];
    if (sqlite3_open(dbpath, &database) == SQLITE_OK)
    {
        NSString *querySQL = [NSString stringWithFormat:
        @"select name, department, year from studentsDetail where
        regno=\"%@\"",registerNumber];
        const char *query_stmt = [querySQL UTF8String];
        NSMutableArray *resultArray = [[NSMutableArray alloc]init];
        if (sqlite3_prepare_v2(database,
           query_stmt, -1, &statement, NULL) == SQLITE_OK)
        {
            if (sqlite3_step(statement) == SQLITE_ROW)
            {
                NSString *name = [[NSString alloc] initWithUTF8Stri
                 (const char *) sqlite3_column_text(statement, 0)];
                [resultArray addObject:name];
                NSString *department = [[NSString alloc] initWithUT
                (const char *) sqlite3_column_text(statement, 1)];
                [resultArray addObject:department];
                NSString *year = [[NSString alloc]initWithUTF8Strin
                (const char *) sqlite3_column_text(statement, 2)];
                [resultArray addObject:year];
                return resultArray;
            }
            else{
                NSLog(@"Not found");
                return nil;
            }
            sqlite3_reset(statement);
        }
    }
    return nil;
}
```



8、 如图所示，更新ViewController.xib文件

9、 为上述文本字段创建IBOutlets

10、 为上述按钮创建IBAction

11、 如下所示，更新ViewController.h

```
#import <UIKit/UIKit.h>
#import "DBManager.h"

@interface ViewController : UIViewController<UITextFieldDelegate>
{
    IBOutlet UITextField *regNoTextField;
    IBOutlet UITextField *nameTextField;
    IBOutlet UITextField *departmentTextField;
    IBOutlet UITextField *yearTextField;
    IBOutlet UITextField *findByRegisterNumberTextField;
    IBOutlet UIScrollView *myScrollView;
}

-(IBAction)saveData:(id)sender;
-(IBAction)findData:(id)sender;

@end
```

12、更新ViewController.m,如下所示

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)
  nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNi
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view from its nib
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
```

```
-(IBAction)saveData:(id)sender{
    BOOL success = NO;
    NSString *alertString = @"Data Insertion failed";
    if (regNoTextField.text.length>0 &&nameTextField.text.length>0
    departmentTextField.text.length>0 &&yearTextField.text.length>0
    {
        success = [[DBManager getSharedInstance]saveData:
        regNoTextField.text name:nameTextField.text department:
        departmentTextField.text year:yearTextField.text];
    }
    else{
        alertString = @"Enter all fields";
    }
    if (success == NO) {
        UIAlertView *alert = [[UIAlertView alloc]initWithTitle:
        alertString message:nil
        delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:nil]
        [alert show];
    }
}

-(IBAction)findData:(id)sender{
    NSArray *data = [[DBManager getSharedInstance]findByRegisterNum
    findByRegisterNumberTextField.text];
    if (data == nil) {
        UIAlertView *alert = [[UIAlertView alloc]initWithTitle:
        @"Data not found" message:nil delegate:nil cancelButtonTitl
        @"OK" otherButtonTitles:nil];
        [alert show];
        regNoTextField.text = @"";
        nameTextField.text =@"";
        departmentTextField.text = @"";
        yearTextField.text =@"";
    }
    else{
        regNoTextField.text = findByRegisterNumberTextField.text;
        nameTextField.text =[data objectAtIndex:0];
        departmentTextField.text = [data objectAtIndex:1];
        yearTextField.text =[data objectAtIndex:2];
    }
}

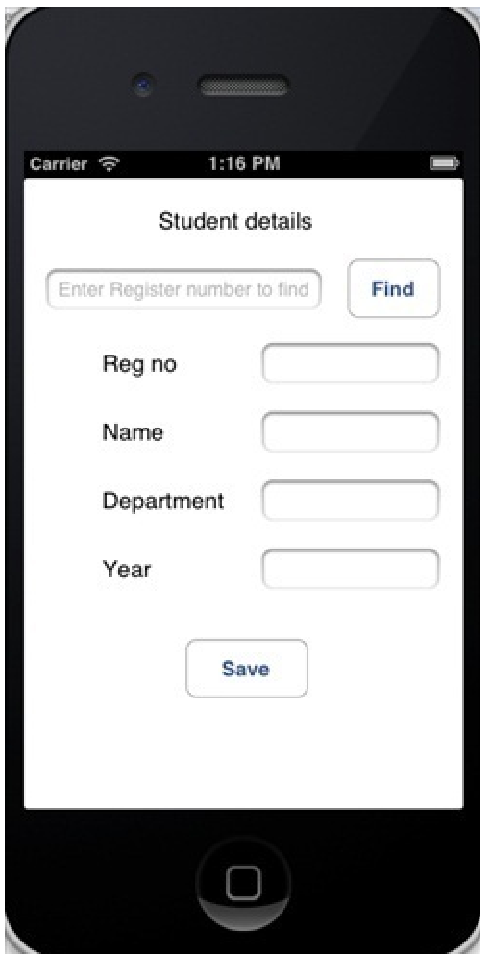
#pragma mark - Text field delegate
-(void)textFieldDidBeginEditing:(UITextField *)textField{
    [myScrollView setFrame:CGRectMake(10, 50, 300, 200)];
    [myScrollView setContentSize:CGSizeMake(300, 350)];
}
-(void)textFieldDidEndEditing:(UITextField *)textField{
    [myScrollView setFrame:CGRectMake(10, 50, 300, 350)];

}
-(BOOL) textFieldShouldReturn:(UITextField *)textField{
```

```
    [textField resignFirstResponder];
    return YES;
}
@end
```

输出

现在当我们运行应用程序时，我们就会获得下面的输出，我们可以在其中添加及查找学生的详细信息



IOS发送电子邮件

简介

我们可以使用IOS设备中的电子邮件应用程序发送电子邮件。

实例步骤

1、创建一个简单的View based application

2、选择项目文件，然后选择目标，然后添加MessageUI.framework

3、在ViewController.xib中添加一个按钮，创建用于发送电子邮件的操作（action）

4、更新ViewController.h,如下所示

```
#import <UIKit/UIKit.h>
#import <MessageUI/MessageUI.h>

@interface ViewController : UIViewController<MFMailComposeViewContr
{
    MFMailComposeViewController *mailComposer;
}

-(IBAction)sendMail:(id)sender;

@end
```

5、如下所示，更新ViewController.m

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

-(void)sendMail:(id)sender{
    mailComposer = [[MFMailComposeViewController alloc]init];
    mailComposer.mailComposeDelegate = self;
    [mailComposer setSubject:@"Test mail"];
    [mailComposer setMessageBody:@"Testing message
    for the test mail" isHTML:NO];
    [self presentModalViewController:mailComposer animated:YES];
}

#pragma mark - mail compose delegate
-(void)mailComposeController:(MFMailComposeViewController *)control
 didFinishWithResult:(MFMailComposeResult)result error:(NSError *)e
   if (result) {
        NSLog(@"Result : %d",result);
    }
    if (error) {
        NSLog(@"Error : %@",error);
    }
    [self dismissModalViewControllerAnimated:YES];

}

@end
```

输出

当运行该应用程序，会看如下的输出结果



当点击"send email"发送按钮后，可以看到如下结果：


Carrier
4:41 PM
Cancel
Test mail
Send
To:
Cc/Bcc:
Subject: Test mail
Testing message for the test mail
Sent from my iPhone Simulator

IOS音频和视频(Audio & Video)

简介

音频和视频在最新的设备中颇为常见。

将iosAVFoundation.framework和MediaPlayer.framework添加到Xcode项目中，可以让IOS支持音频和视频(Audio & Video)。

实例步骤

1、创建一个简单的View based application

2、选择项目文件、选择目标，然后添加AVFoundation.framework和MediaPlayer.framework

3、在ViewController.xib中添加两个按钮，创建一个用于分别播放音频和视频的动作（action）

4、更新ViewController.h,如下所示

```
#import <UIKit/UIKit.h>
#import <AVFoundation/AVFoundation.h>
#import <MediaPlayer/MediaPlayer.h>

@interface ViewController : UIViewController
{
    AVAudioPlayer *audioPlayer;
    MPMoviePlayerViewController *moviePlayer;

}
-(IBAction)playAudio:(id)sender;
-(IBAction)playVideo:(id)sender;
@end
```

5、更新ViewController.m，如下所示

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
}

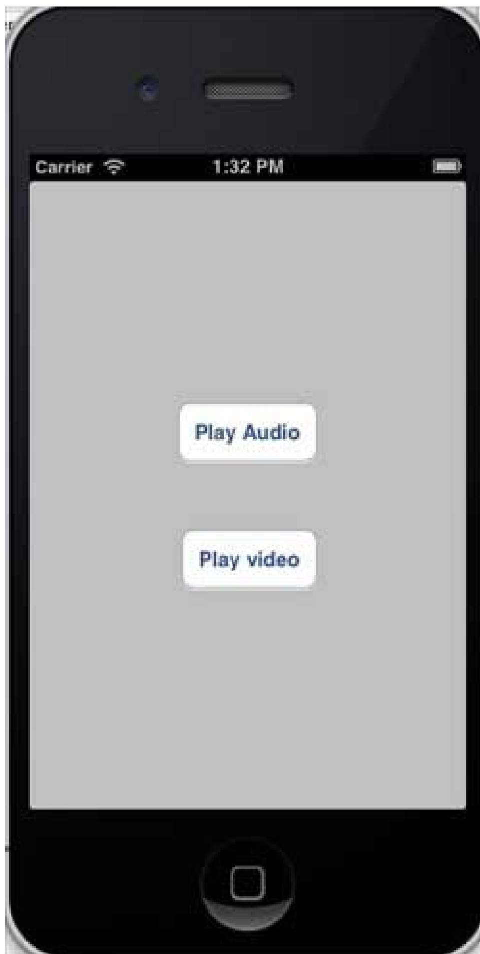
- (void)didReceiveMemoryWarning
{
   [super didReceiveMemoryWarning];
   // Dispose of any resources that can be recreated.
}
-(IBAction)playAudio:(id)sender{
   NSString *path = [[NSBundle mainBundle]
   pathForResource:@"audioTest" ofType:@"mp3"];
   audioPlayer = [[AVAudioPlayer alloc]initWithContentsOfURL:
   [NSURL fileURLWithPath:path] error:NULL];
   [audioPlayer play];
}
-(IBAction)playVideo:(id)sender{
   NSString *path = [[NSBundle mainBundle]pathForResource:
   @"videoTest" ofType:@"mov"];
   moviePlayer = [[MPMoviePlayerViewController
   alloc]initWithContentURL:[NSURL fileURLWithPath:path]];
   [self presentModalViewController:moviePlayer animated:NO];
}
@end
```

注意项

需要添加音频和视频文件，以确保获得预期的输出

输出

运行该程序，得到的输出结果如下



当我们点击 play video(播放视频)显示如下：


Carrier
1:32 PM
Done
0:00
-0:05

IOS文件处理

简介

文件处理不能直观的通过应用程序来解释，我们可以从以下实例来了解IOS的文件处理。

IOS中对文件的操作. 因为应用是在沙箱（sandbox）中的，在文件读写权限上受到限制，只能在几个目录下读写文件。

文件处理中使用的方法

下面列出了用于访问和操作文件的方法的列表。

以下实例你必须替换FilePath1、FilePath和FilePath字符串为完整的文件路径，以获得所需的操作。

检查文件是否存在

```
NSFileManager *fileManager = [NSFileManager defaultManager];
//Get documents directory
NSArray *directoryPaths = NSSearchPathForDirectoriesInDomains
(NSDocumentDirectory, NSUserDomainMask, YES);
NSString *documentsDirectoryPath = [directoryPaths objectAtInde
if ([fileManager fileExistsAtPath:@""]==YES) {
     NSLog(@"File exists");
 }
```

比较两个文件的内容

```
if ([fileManager contentsEqualAtPath:@"FilePath1" andPath:@" Fi
   NSLog(@"Same content");
}
```

检查是否可写、可读、可执行文件

```
    if ([fileManager isWritableFileAtPath:@"FilePath"]) {
        NSLog(@"isWritable");
     }
     if ([fileManager isReadableFileAtPath:@"FilePath"]) {
        NSLog(@"isReadable");
     }
     if ( [fileManager isExecutableFileAtPath:@"FilePath"]){
        NSLog(@"is Executable");
     }
```

移动文件

```
     if([fileManager moveItemAtPath:@"FilePath1"
     toPath:@"FilePath2" error:NULL]){
        NSLog(@"Moved successfully");
     }
```

复制文件

```
     if ([fileManager copyItemAtPath:@"FilePath1"
     toPath:@"FilePath2"  error:NULL]) {
        NSLog(@"Copied successfully");
     }
```

删除文件

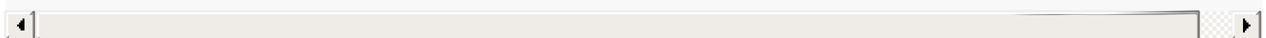
```
   if ([fileManager removeItemAtPath:@"FilePath" error:NULL]) {
        NSLog(@"Removed successfully");
     }
```

读取文件

```
   NSData *data = [fileManager contentsAtPath:@"Path"];
```

写入文件

```
    [fileManager createFileAtPath:@"" contents:data attributes:nil];
```



IOS地图开发

简介

IOS地图帮助我们定位位置，IOS地图使用 MapKit 框架。

实例步骤

1.创建一个简单的 View based application

2.选择项目文件，然后选择目标，然后添加MapKit.framework.

3.添加 Corelocation.framework

4.向 ViewController.xib 添加地图查看和创建 ibOutlet 并且命名为mapView。

5.通过"File-> New -> File... -> "选择 Objective C class创建一个新的文件，单击下一步

6."sub class of"为 NSObject，类作命名为MapAnnotation

7.选择创建

8.更新MapAnnotation.h，如下所示

```
#import <Foundation/Foundation.h>
#import <MapKit/MapKit.h>

@interface MapAnnotation : NSObject<MKAnnotation>
@property (nonatomic, strong) NSString *title;
@property (nonatomic, readwrite) CLLocationCoordinate2D coordinate;

- (id)initWithTitle:(NSString *)title andCoordinate:
  (CLLocationCoordinate2D)coordinate2d;

@end
```

9.更新MapAnnotation.m，如下所示

```
#import "MapAnnotation.h"

@implementation MapAnnotation
-(id)initWithTitle:(NSString *)title andCoordinate:
 (CLLocationCoordinate2D)coordinate2d{
    self.title = title;
    self.coordinate =coordinate2d;
    return self;
}
@end
```

10.更新ViewController.h， 如下所示

```
#import <UIKit/UIKit.h>
#import <MapKit/MapKit.h>
#import <CoreLocation/CoreLocation.h>
@interface ViewController : UIViewController<MKMapViewDelegate>
{
    MKMapView *mapView;
}
@end
```

11.更新ViewController.m， 如下所示

```
#import "ViewController.h"
#import "MapAnnotation.h"

@interface ViewController ()

@end

@implementation ViewController

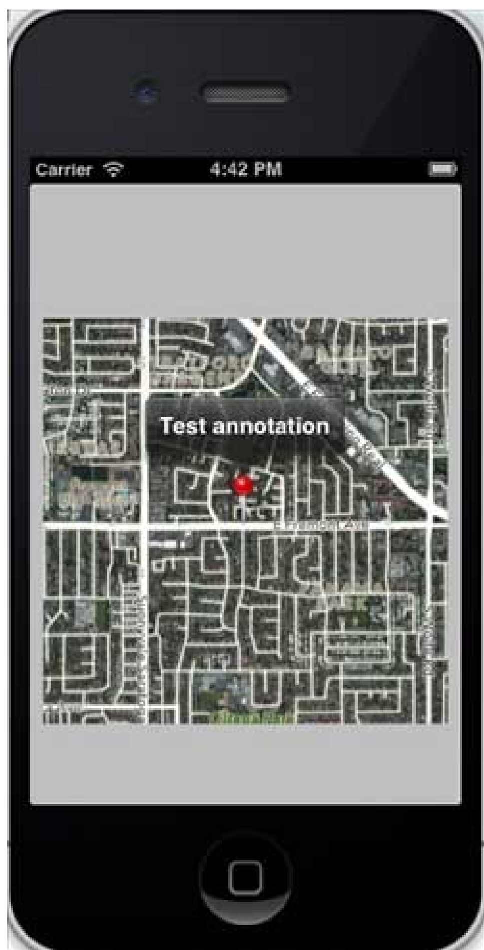
- (void)viewDidLoad
{
   [super viewDidLoad];
   mapView = [[MKMapView alloc]initWithFrame:
   CGRectMake(10, 100, 300, 300)];
   mapView.delegate = self;
   mapView.centerCoordinate = CLLocationCoordinate2DMake(37.32, -12
   mapView.mapType = MKMapTypeHybrid;
   CLLocationCoordinate2D location;
   location.latitude = (double) 37.332768;
   location.longitude = (double) -122.030039;
   // Add the annotation to our map view
   MapAnnotation *newAnnotation = [[MapAnnotation alloc]
   initWithTitle:@"Apple Head quaters" andCoordinate:location];
   [mapView addAnnotation:newAnnotation];
   CLLocationCoordinate2D location2;
   location2.latitude = (double) 37.35239;
   location2.longitude = (double) -122.025919;
   MapAnnotation *newAnnotation2 = [[MapAnnotation alloc]
   initWithTitle:@"Test annotation" andCoordinate:location2];
   [mapView addAnnotation:newAnnotation2];
   [self.view addSubview:mapView];
}
// When a map annotation point is added, zoom to it (1500 range)
- (void)mapView:(MKMapView *)mv didAddAnnotationViews:(NSArray *)vi
{
   MKAnnotationView *annotationView = [views objectAtIndex:0];
   id <MKAnnotation> mp = [annotationView annotation];
   MKCoordinateRegion region = MKCoordinateRegionMakeWithDistance
   ([mp coordinate], 1500, 1500);
   [mv setRegion:region animated:YES];
   [mv selectAnnotation:mp animated:YES];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

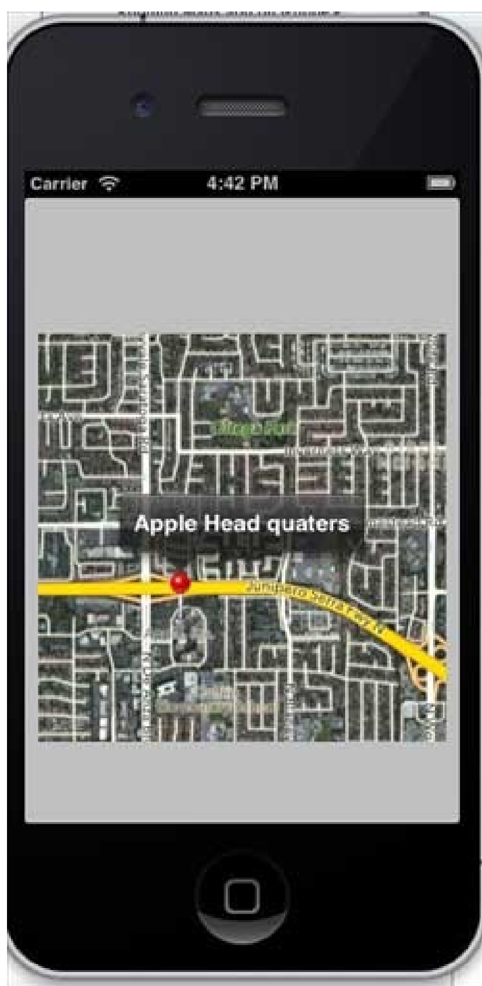
@end
```

输出

运行应用程序时，输出结果如下



当我们向上滚动地图时，输出结果如下


Carrier
4:42 PM
Apple Head quaters
Junipero Serra Fwy N

IOS应用内购买

简介

应用程序内购买是应用程序用于购买额外内容或升级功能。

实例步骤

1.在 iTunes 连接中请确保拥有一个唯一的 App ID（unique App ID ），当创建捆绑的ID（ bundle ID）应用程序更新时，代码会以相应的配置文件签名在Xcode上

2.创建新的应用程序和更新应用程序信息。你可以知道更多有关的，在苹果的 添加新的应用程序 文档中

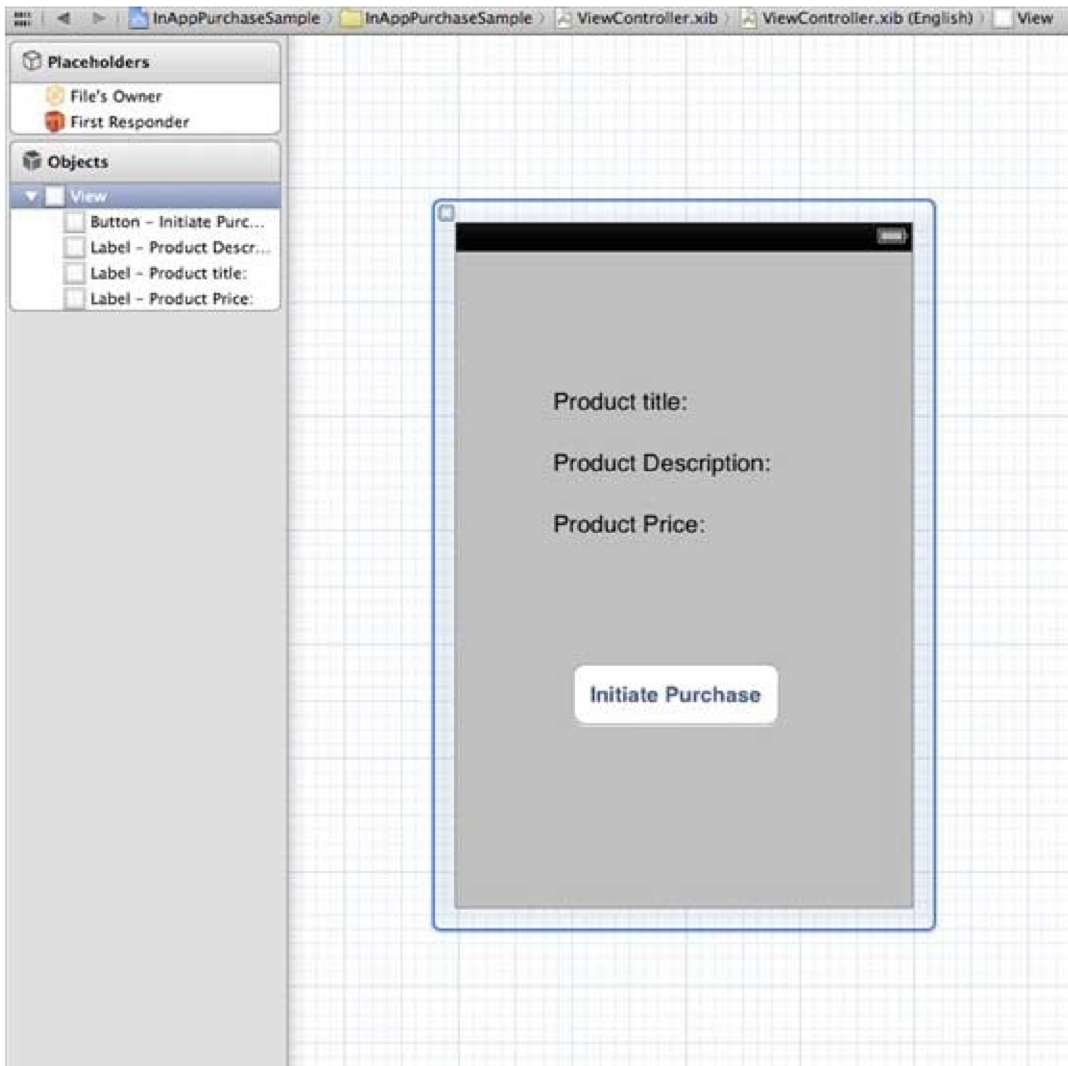
3.在应用程序页的管理应用程序(Manage In-App Purchase)中，为app内付费添加新产品

4.确保设置的应用程序为的银行详细。需要将其设置为在应用程序内购买（In-App purchase）。此外在 iTunes 中使用管理用户（Manage Users）选项，创建一个测试用户帐户连接您的应用程序的页。

5.下一步是与处理代码和为我们在应用程序内购买创建有关的 UI。

6.创建一个单一的视图应用程序，并在 iTunes 中指定的标识符连接输入捆绑标识符

7.更新ViewController.xib ，如下所示



8.为三个标签创建IBOutlets，且将按钮分别命名为 productTitleLabel、productDescriptionLabel、 productPriceLabel 和 purchaseButton

9.选择项目文件，然后选择目标，然后添加StoreKit.framework

10.更新ViewController.h， 如下所示

```
#import <UIKit/UIKit.h>
#import <StoreKit/StoreKit.h>

@interface ViewController : UIViewController<
SKProductsRequestDelegate,SKPaymentTransactionObserver>
{
    SKProductsRequest *productsRequest;
    NSArray *validProducts;
    UIActivityIndicatorView *activityIndicatorView;
    IBOutlet UILabel *productTitleLabel;
    IBOutlet UILabel *productDescriptionLabel;
    IBOutlet UILabel *productPriceLabel;
    IBOutlet UIButton *purchaseButton;
}
- (void)fetchAvailableProducts;
- (BOOL)canMakePurchases;
- (void)purchaseMyProduct:(SKProduct*)product;
- (IBAction)purchase:(id)sender;

@end
```

11.更新ViewController.m， 如下所示

```
#import "ViewController.h"
#define kTutorialPointProductID
@"com.tutorialPoints.testApp.testProduct"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Adding activity indicator
    activityIndicatorView = [[UIActivityIndicatorView alloc]
    initWithActivityIndicatorStyle:UIActivityIndicatorViewStyleWhit
    activityIndicatorView.center = self.view.center;
    [activityIndicatorView hidesWhenStopped];
    [self.view addSubview:activityIndicatorView];
    [activityIndicatorView startAnimating];
    //Hide purchase button initially
    purchaseButton.hidden = YES;
    [self fetchAvailableProducts];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
```

```
    // Dispose of any resources that can be recreated.
}

-(void)fetchAvailableProducts{
    NSSet *productIdentifiers = [NSSet
    setWithObjects:kTutorialPointProductID,nil];
    productsRequest = [[SKProductsRequest alloc]
    initWithProductIdentifiers:productIdentifiers];
    productsRequest.delegate = self;
    [productsRequest start];
}

- (BOOL)canMakePurchases
{
    return [SKPaymentQueue canMakePayments];
}
- (void)purchaseMyProduct:(SKProduct*)product{
    if ([self canMakePurchases]) {
        SKPayment *payment = [SKPayment paymentWithProduct:product]
        [[SKPaymentQueue defaultQueue] addTransactionObserver:self]
        [[SKPaymentQueue defaultQueue] addPayment:payment];
    }
    else{
        UIAlertView *alertView = [[UIAlertView alloc]initWithTitle:
        @"Purchases are disabled in your device" message:nil delega
        self cancelButtonTitle:@"Ok" otherButtonTitles: nil];
        [alertView show];
    }
}
-(IBAction)purchase:(id)sender{
    [self purchaseMyProduct:[validProducts objectAtIndex:0]];
    purchaseButton.enabled = NO;
}

#pragma mark StoreKit Delegate

-(void)paymentQueue:(SKPaymentQueue *)queue
 updatedTransactions:(NSArray *)transactions {
    for (SKPaymentTransaction *transaction in transactions) {
        switch (transaction.transactionState) {
            case SKPaymentTransactionStatePurchasing:
                    NSLog(@"Purchasing");
                break;
            case SKPaymentTransactionStatePurchased:
               if ([transaction.payment.productIdentifier
                  isEqualToString:kTutorialPointProductID]) {
                   NSLog(@"Purchased ");
                   UIAlertView *alertView = [[UIAlertView alloc]ini
                   @"Purchase is completed succesfully" message:nil
                   self cancelButtonTitle:@"Ok" otherButtonTitles:
                   [alertView show];
                }
                [[SKPaymentQueue defaultQueue] finishTransaction:tr
```

```
                break;
            case SKPaymentTransactionStateRestored:
                NSLog(@"Restored ");
                [[SKPaymentQueue defaultQueue] finishTransaction:tr
                break;
            case SKPaymentTransactionStateFailed:
                NSLog(@"Purchase failed ");
                break;
            default:
                break;
        }
    }
}

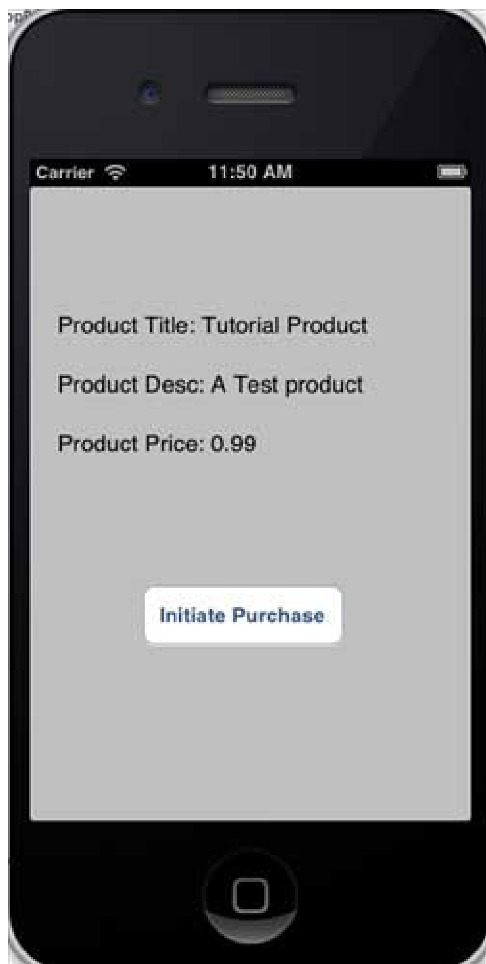
-(void)productsRequest:(SKProductsRequest *)request
 didReceiveResponse:(SKProductsResponse *)response
{
    SKProduct *validProduct = nil;
    int count = [response.products count];
    if (count>0) {
        validProducts = response.products;
        validProduct = [response.products objectAtIndex:0];
        if ([validProduct.productIdentifier
           isEqualToString:kTutorialPointProductID]) {
            [productTitleLabel setText:[NSString stringWithFormat:
            @"Product Title: %@",validProduct.localizedTitle]];
            [productDescriptionLabel setText:[NSString stringWithFo
            @"Product Desc: %@",validProduct.localizedDescription]]
            [productPriceLabel setText:[NSString stringWithFormat:
            @"Product Price: %@",validProduct.price]];
        }
    } else {
        UIAlertView *tmp = [[UIAlertView alloc]
                            initWithTitle:@"Not Available"
                            message:@"No products to purchase"
                            delegate:self
                            cancelButtonTitle:nil
                            otherButtonTitles:@"Ok", nil];
        [tmp show];
    }
    [activityIndicatorView stopAnimating];
    purchaseButton.hidden = NO;
}

@end
```

注意： 需要修改你创建In-App Pur（应用内购买）的 kTutorialPointProductID 。通过修改fetchAvailableProducts产品标识符的 NSSet, 你可以添加多个产品。

输出

运行该应用程序,输出结果如下



确保已经中登录。单击购买选择现有的Apple ID。输入有效的测试帐户的用户名和密码。几秒钟后，显示下面的信息



一旦产品成功购买，将获得以下信息。可以在显示此信息的地方，更新应用功能相关的代码


Carrier
11:55 AM
Product Title: Tutorial Product
Product Desc: A Test product
Purchase is completed succesfully
Ok
Initiate Purchase

IOS iAD整合

简介

IAD是苹果推出的广告平台，它可以帮助开发者从应用程序中获取收入。

实例步骤

1. 创建一个简单的View based application

2. 选择项目文件，然后选择目标，然后选择框架并添加 iAd.framework。

3. 更新 ViewController.h 如下所示

```
#import <UIKit/UIKit.h>
#import <iAd/iAd.h>
@interface ViewController : UIViewController<ADBannerViewDelegate>
{
    ADBannerView *bannerView;
}
@end
```

4. 更新ViewController.m， 如下所示

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    bannerView = [[ADBannerView alloc]initWithFrame:
    CGRectMake(0, 0, 320, 50)];
    // Optional to set background color to clear color
    [bannerView setBackgroundColor:[UIColor clearColor]];
    [self.view addSubview: bannerView];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

#pragma mark - AdViewDelegates

-(void)bannerView:(ADBannerView *)banner
 didFailToReceiveAdWithError:(NSError *)error{
    NSLog(@"Error loading");
}

-(void)bannerViewDidLoadAd:(ADBannerView *)banner{
    NSLog(@"Ad loaded");
}
-(void)bannerViewWillLoadAd:(ADBannerView *)banner{
    NSLog(@"Ad will load");
}
-(void)bannerViewActionDidFinish:(ADBannerView *)banner{
    NSLog(@"Ad did finish");

}
@end
```

输出

运行该应用程序,得到如下输出结果:


Carrier
4:28 PM
$1234
Easily generate
more revenue from
your application.
Find Out How
iAd

IOS GameKit

简介

GameKit是iOS SDK中一个常用的框架。其核心功能有3个：

- 交互游戏平台Game Center,
- P2P设备通讯功能
- In-Game Voice。

实例步骤

1.在链接 iTunes 时请确保拥有一个唯一的 App ID（ unique App ID），App ID在我们应用程序更新 bundle ID时及在Xcode代码签名与相应的配置文件需要使用到。

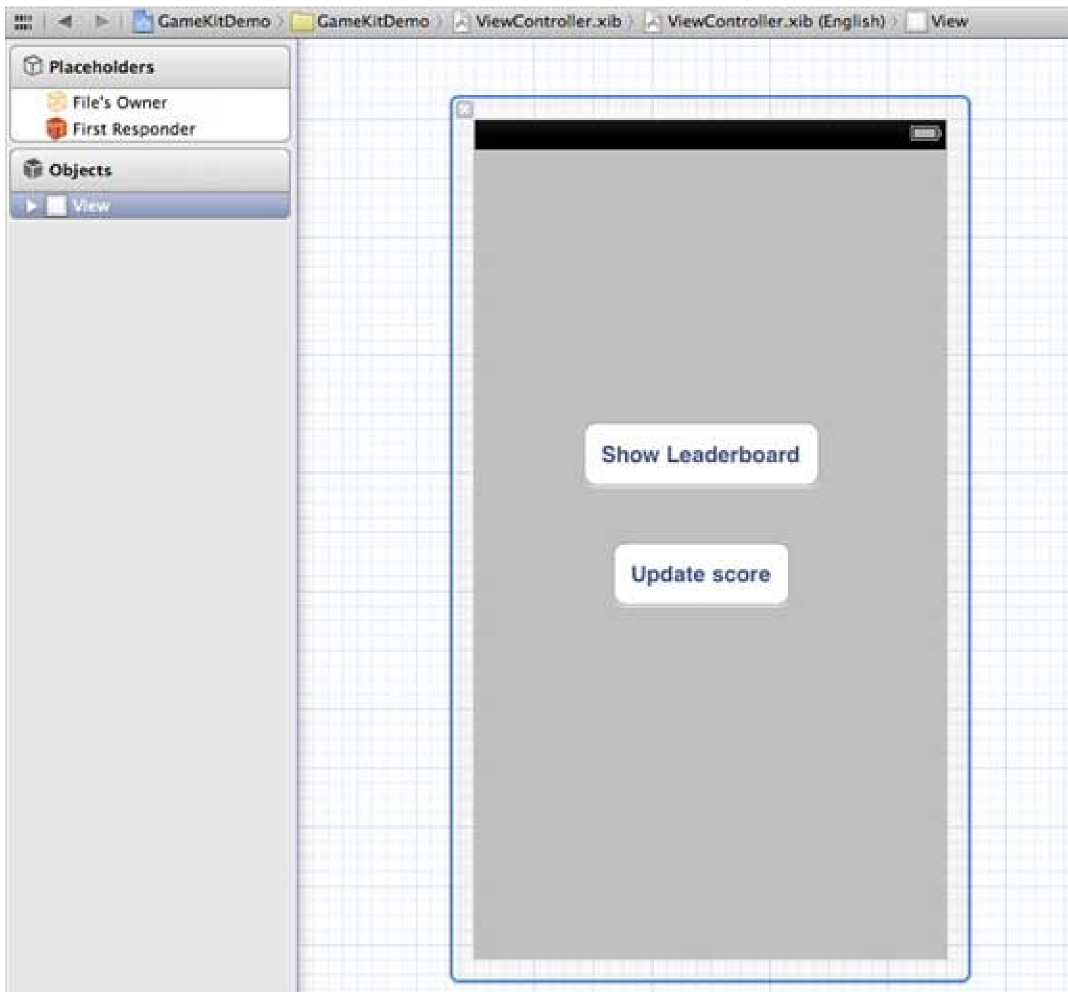
2.创建新的应用程序和更新应用程序信息。在添加新的应用程序文档可以了解更多有关信息。

3.打开你申请的application,点击Manage Game Center选项。进入后点击Enable Game Center使你的Game Center生效。接下来设置自己的Leaderboard和Achievements。

4.下一步涉及处理代码，并为我们的应用程序创建用户界面。

5.创建一个single view application，并输入 bundle identifier 。

6.更新 ViewController.xib，如下所示



7.选择项目文件，然后选择目标，然后添加GameKit.framework

8.为已添加的按钮创建IBActions

9.更新ViewController.h文件，如下所示

```
#import <UIKit/UIKit.h>
#import <GameKit/GameKit.h>

@interface ViewController : UIViewController
<GKLeaderboardViewControllerDelegate>

-(IBAction)updateScore:(id)sender;
-(IBAction)showLeaderBoard:(id)sender;

@end
```

10.更新ViewController.m，如下所示

```
#import "ViewController.h"

@interface ViewController ()
```

```
@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    if([GKLocalPlayer localPlayer].authenticated == NO)
    {
      [[GKLocalPlayer localPlayer]
      authenticateWithCompletionHandler:^(NSError *error)
      {
         NSLog(@"Error%@",error);
      }];
    }
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
- (void) updateScore: (int64_t) score
forLeaderboardID: (NSString*) category
{
    GKScore *scoreObj = [[GKScore alloc]
    initWithCategory:category];
    scoreObj.value = score;
    scoreObj.context = 0;
    [scoreObj reportScoreWithCompletionHandler:^(NSError *error) {
        // Completion code can be added here
        UIAlertView *alert = [[UIAlertView alloc]
        initWithTitle:nil message:@"Score Updated Succesfully"
        delegate:self cancelButtonTitle:@"Ok" otherButtonTitles: ni
        [alert show];

    }];
}
-(IBAction)updateScore:(id)sender{
    [self updateScore:200 forLeaderboardID:@"tutorialsPoint"];
}
-(IBAction)showLeaderBoard:(id)sender{
    GKLeaderboardViewController *leaderboardViewController =
    [[GKLeaderboardViewController alloc] init];
    leaderboardViewController.leaderboardDelegate = self;
    [self presentModalViewController:
    leaderboardViewController animated:YES];

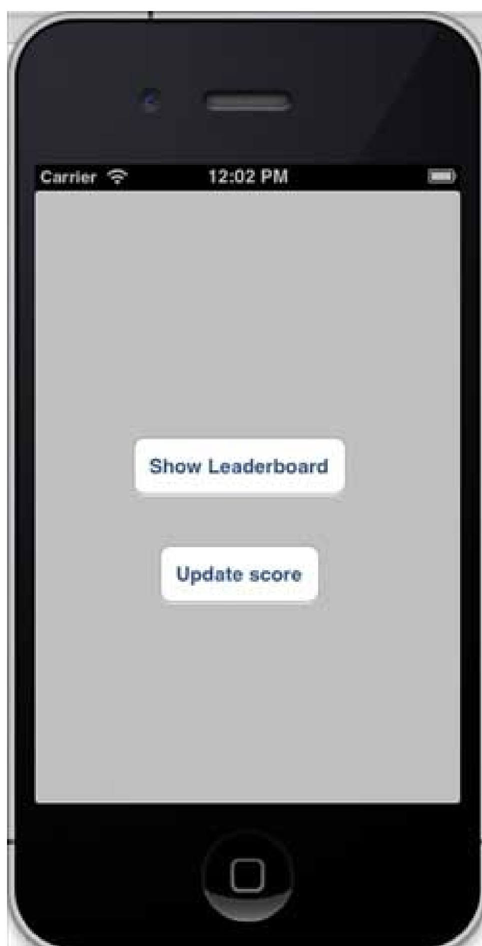
}
#pragma mark - Gamekit delegates
- (void)leaderboardViewControllerDidFinish:
(GKLeaderboardViewController *)viewController{
    [self dismissModalViewControllerAnimated:YES];
```

```
}

@end
```

输出

运行该应用程序，输出结果如下



当我们单击显示排行榜时，屏幕显示如下:



当我们点击更新分数，比分将被更新到我们排行榜上，我们会得到一个信息，如下图所示


Carrier
12:05 PM
Score Updated Succesfully
Ok
Update score

IOS 故事板(Storyboards)

简介

Storyboards在 iOS 5 中才有介绍，当我们用Storyboards时，部署目标应该是iOS5.0或更高版本。

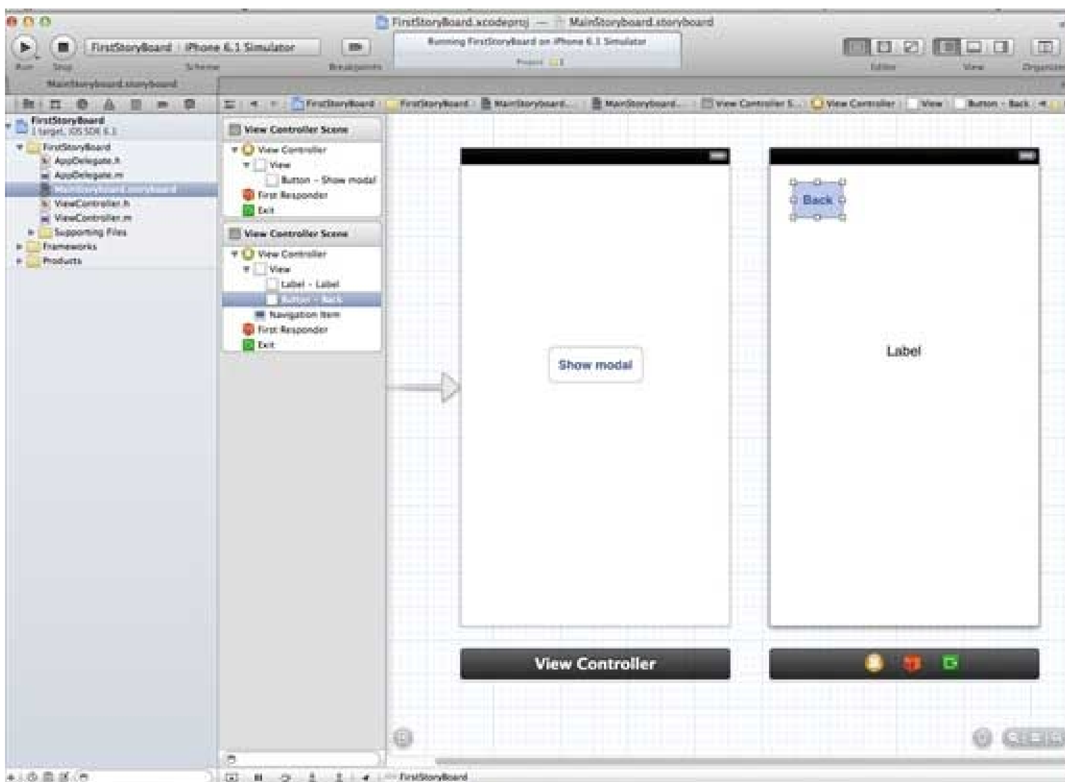
Storyboards 帮助我们了解视觉流动的画面，在界面为

MainStoryboard.storyboard下创建所有应用程序屏幕。

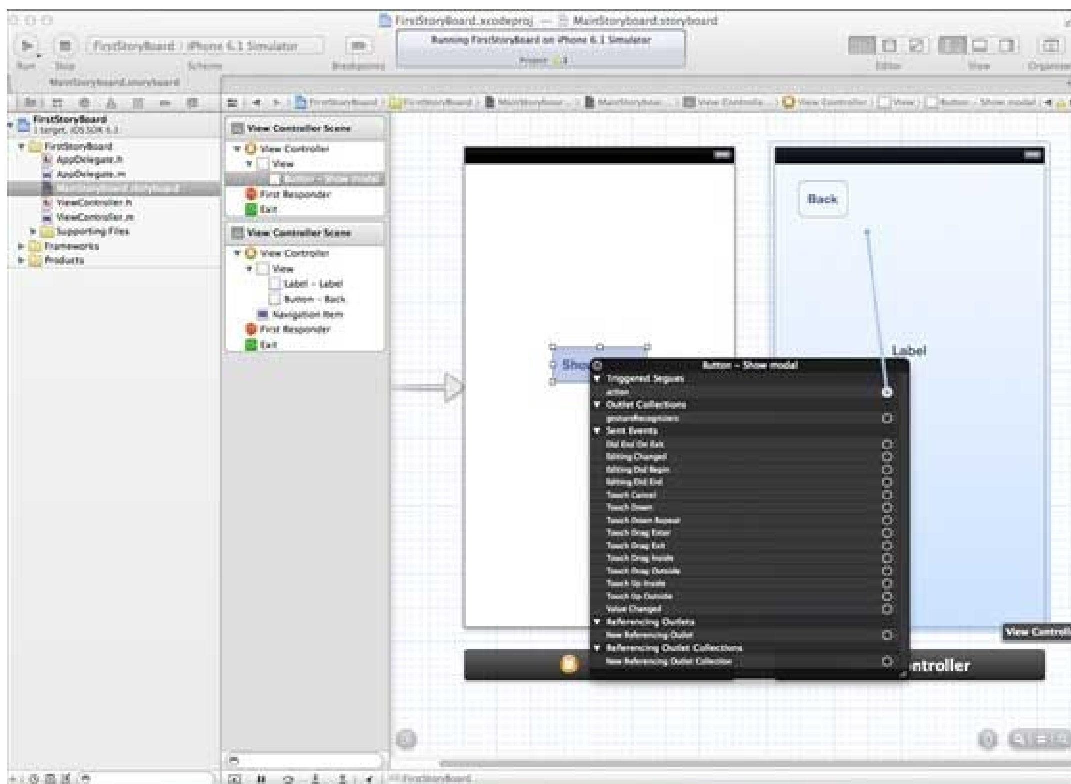
实例步骤

1. 创建一个single view application，创建应用程序时选择 storyboard 复选框。

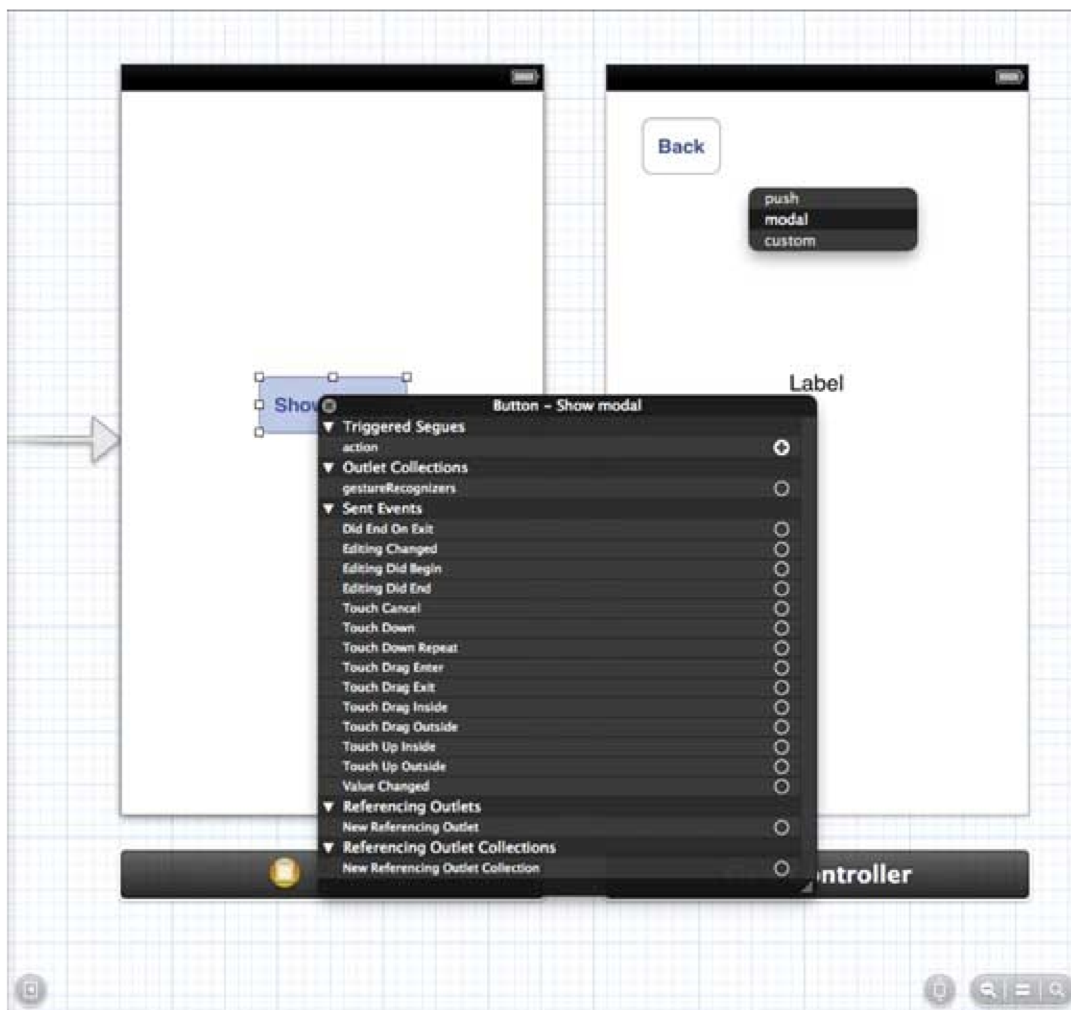
2. 选择MainStoryboard.storyboard，在这里你可以找到单一视图控制器。添加一个视图控制器，更新视图控制器，如下所示



3.连接两个视图控制器。右键单击"show modal（显示模式）"按钮，在左侧视图控制器将其拖动到右视视图控制器中,如下图所示：



4.现在从如下所示的三个显示选项中选择modal(模态)



5.更新 ViewController.h 如下所示

```
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController

-(IBAction)done:(UIStoryboardSegue *)seque;

@end
```

6.更新 ViewController.m 如下所示

```
#import "ViewController.h"

@interface ViewController ()

@end

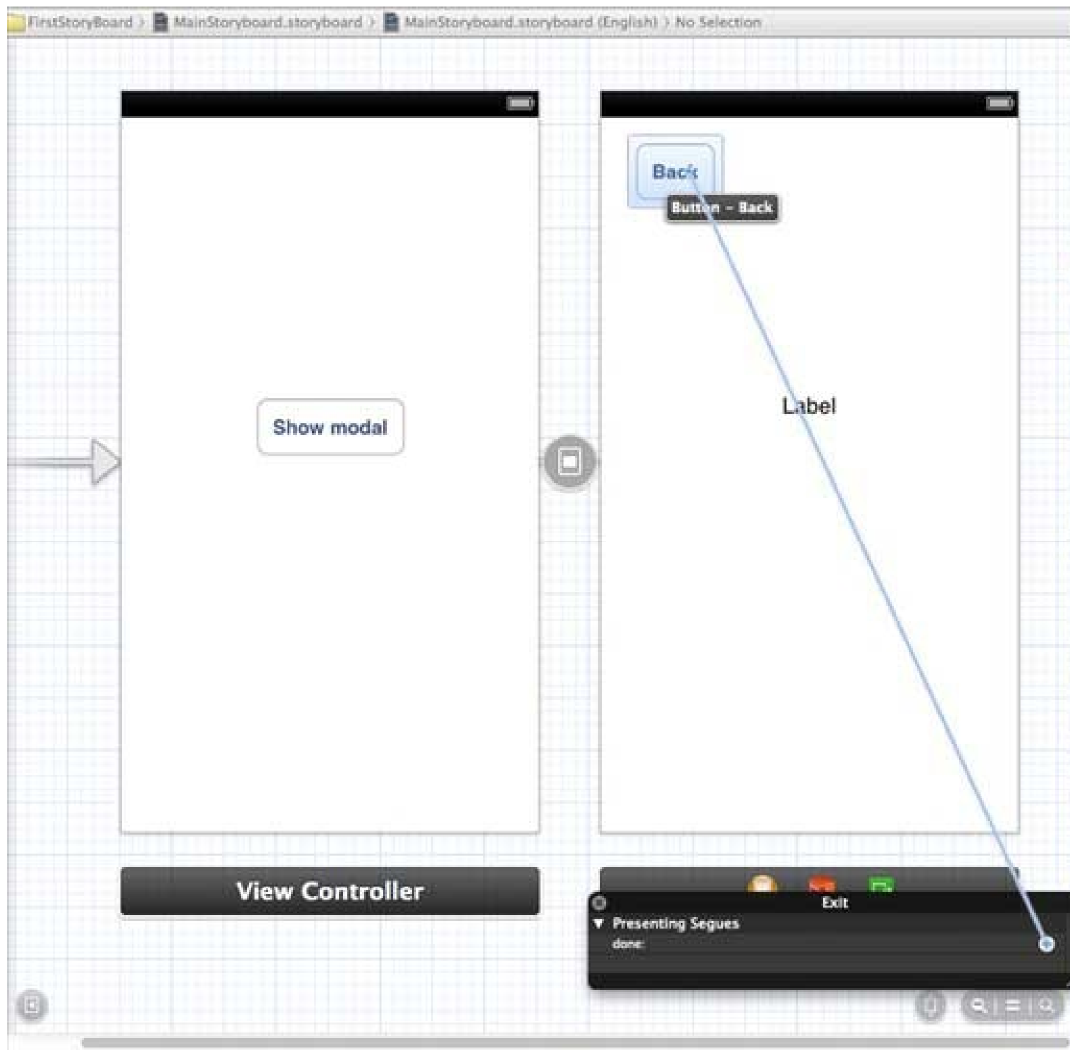
@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
-(IBAction)done:(UIStoryboardSegue *)seque{
    [self.navigationController popViewControllerAnimated:YES];
}

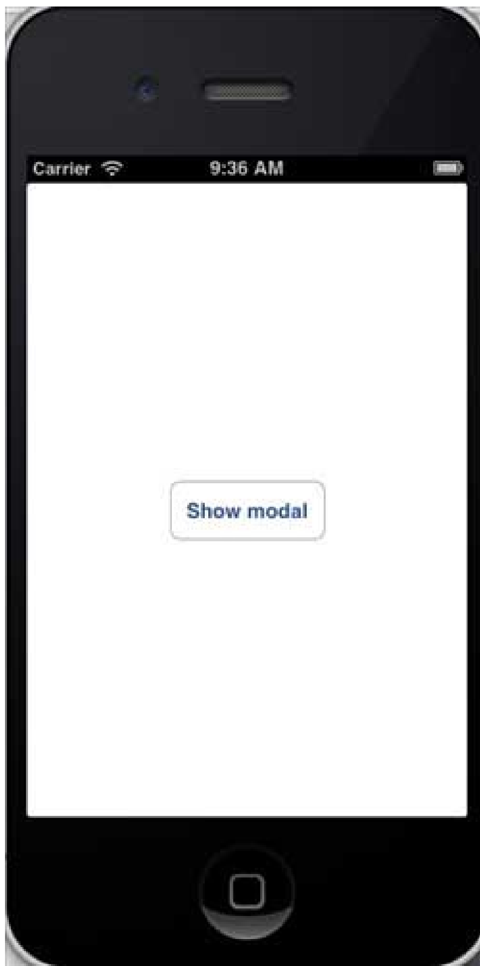
@end
```

7.选择"MainStoryboard.storyboard"，并右键点击"Exit "按钮，在右侧视图控制器中选择和连接后退按钮，如下图所示

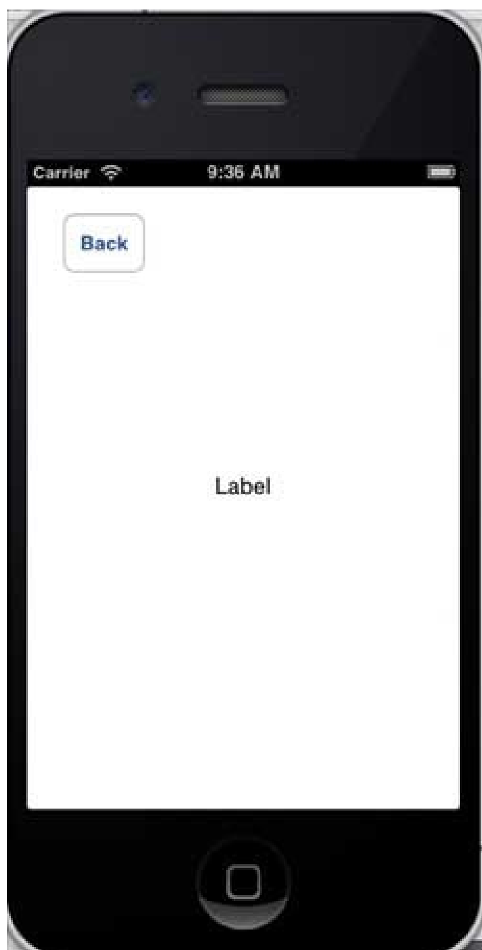


输出

在iPhone设备中运行该应用程序,得到如下输出结果



现在，选择显示模式，将得到下面的输出结果


Carrier
9:36 AM
Back
Label

IOS自动布局

简介

自动布局在iOS 6.0中引入，仅可以支持IOS6.0 及 更高版本。它可以帮助我们创建用于多个种设备的界面。

实例步骤

1.创建一个简单的 View based application

2.修改 ViewController.m 的文件内容，如下所示

```
#import "ViewController.h"
@interface ViewController ()
@property (nonatomic, strong) UIButton *leftButton;
@property (nonatomic, strong) UIButton *rightButton;
@property (nonatomic, strong) UITextField *textfield;

@end
@implementation ViewController

- (void)viewDidLoad{
    [super viewDidLoad];
    UIView *superview = self.view;
    /*1\. Create leftButton and add to our view*/
    self.leftButton = [UIButton buttonWithType:UIButtonTypeRoundedR
    self.leftButton.translatesAutoresizingMaskIntoConstraints = NO;
    [self.leftButton setTitle:@"LeftButton" forState:UIControlState
    [self.view addSubview:self.leftButton];
    /* 2\. Constraint to position LeftButton's X*/
    NSLayoutConstraint *leftButtonXConstraint = [NSLayoutConstraint
    constraintWithItem:self.leftButton attribute:NSLayoutAttributeC
    relatedBy:NSLayoutRelationGreaterThanOrEqual toItem:superview a
    NSLayoutAttributeCenterX multiplier:1.0 constant:-60.0f];
    /* 3\. Constraint to position LeftButton's Y*/
    NSLayoutConstraint *leftButtonYConstraint = [NSLayoutConstraint
    constraintWithItem:self.leftButton attribute:NSLayoutAttributeC
    relatedBy:NSLayoutRelationEqual toItem:superview attribute:
    NSLayoutAttributeCenterY multiplier:1.0f constant:0.0f];
    /* 4\. Add the constraints to button's superview*/
    [superview addConstraints:@[ leftButtonXConstraint,
    leftButtonYConstraint]];
    /*5\. Create rightButton and add to our view*/
    self.rightButton = [UIButton buttonWithType:UIButtonTypeRoundec
    self.rightButton.translatesAutoresizingMaskIntoConstraints = NO
    [self.rightButton setTitle:@"RightButton" forState:UIControlSta
```

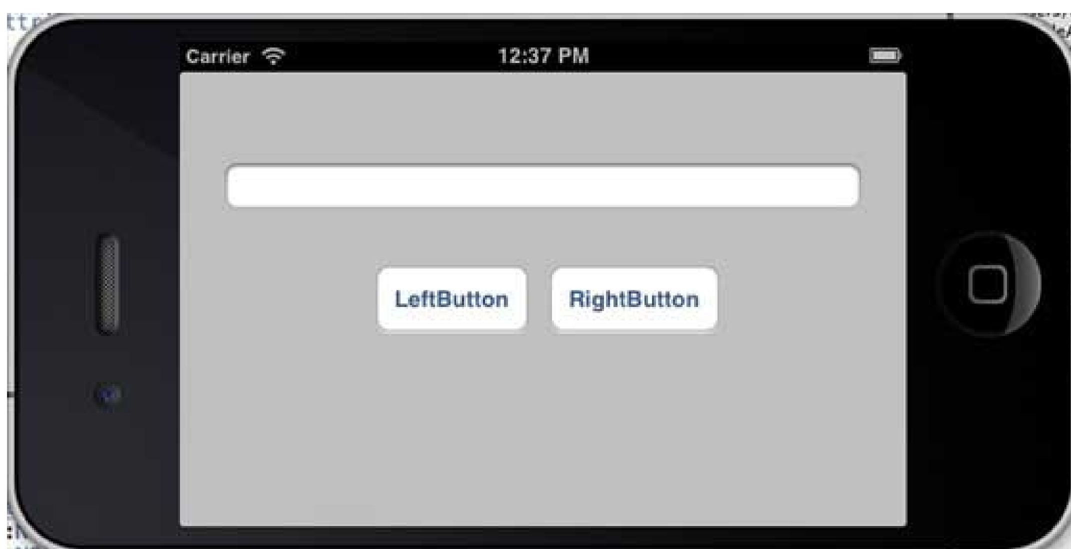
```
    [self.view addSubview:self.rightButton];
    /*6\. Constraint to position RightButton's X*/
    NSLayoutConstraint *rightButtonXConstraint = [NSLayoutConstrain
    constraintWithItem:self.rightButton attribute:NSLayoutAttribute
    relatedBy:NSLayoutRelationGreaterThanOrEqual toItem:superview a
    NSLayoutAttributeCenterX multiplier:1.0 constant:60.0f];
    /*7\. Constraint to position RightButton's Y*/
    rightButtonXConstraint.priority = UILayoutPriorityDefaultHigh;
    NSLayoutConstraint *centerYMyConstraint = [NSLayoutConstraint
    constraintWithItem:self.rightButton attribute:NSLayoutAttribute
    relatedBy:NSLayoutRelationGreaterThanOrEqual toItem:superview a
    NSLayoutAttributeCenterY multiplier:1.0f constant:0.0f];
    [superview addConstraints:@[centerYMyConstraint,
    rightButtonXConstraint]];
   //8\. Add Text field
    self.textfield = [[UITextField alloc]initWithFrame:
    CGRectMake(0, 100, 100, 30)];
    self.textfield.borderStyle = UITextBorderStyleRoundedRect;
    self.textfield.translatesAutoresizingMaskIntoConstraints = NO;
    [self.view addSubview:self.textfield];
    //9\. Text field Constraints
    NSLayoutConstraint *textFieldTopConstraint = [NSLayoutConstrain
    constraintWithItem:self.textfield attribute:NSLayoutAttributeTo
    relatedBy:NSLayoutRelationGreaterThanOrEqual toItem:superview
    attribute:NSLayoutAttributeTop multiplier:1.0 constant:60.0f];
    NSLayoutConstraint *textFieldBottomConstraint = [NSLayoutConstr
    constraintWithItem:self.textfield attribute:NSLayoutAttributeTo
    relatedBy:NSLayoutRelationGreaterThanOrEqual toItem:self.rightB
    attribute:NSLayoutAttributeTop multiplier:0.8 constant:-60.0f];
    NSLayoutConstraint *textFieldLeftConstraint = [NSLayoutConstrai
    constraintWithItem:self.textfield attribute:NSLayoutAttributeLe
    relatedBy:NSLayoutRelationEqual toItem:superview attribute:
    NSLayoutAttributeLeft multiplier:1.0 constant:30.0f];
    NSLayoutConstraint *textFieldRightConstraint = [NSLayoutConstra
    constraintWithItem:self.textfield attribute:NSLayoutAttributeRi
    relatedBy:NSLayoutRelationEqual toItem:superview attribute:
    NSLayoutAttributeRight multiplier:1.0 constant:-30.0f];
    [superview addConstraints:@[textFieldBottomConstraint ,
    textFieldLeftConstraint, textFieldRightConstraint,
    textFieldTopConstraint]];
}
- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}
@end
```

输出

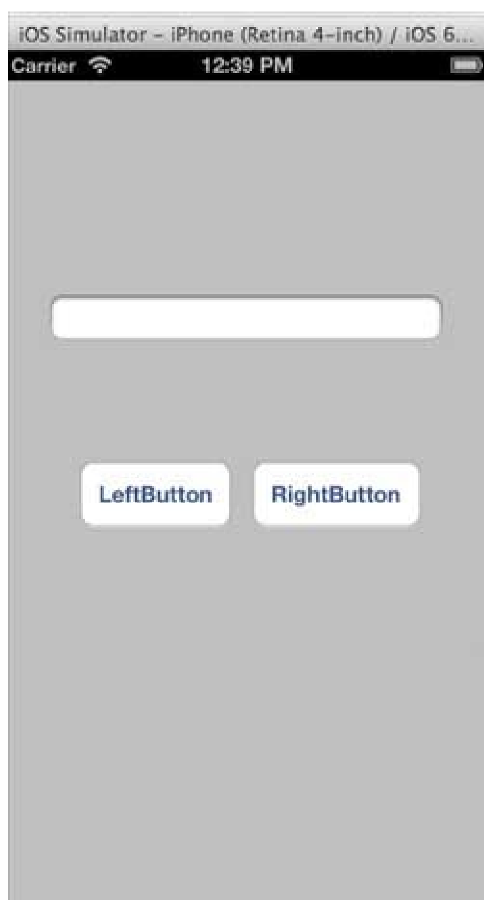
运行应用程序，在 iPhone 模拟器上会有下面的输出结果



当我们更改模拟器为横向的方向时，输出结果如下



我们在 iPhone 5 模拟器上运行同一应用程序时,输出结果如下



当我们更改模拟器为横向的方向时，输出结果如下:



IOS-Twitter和Facebook

简介

Twitter已经整合到iOS5.0，而Facebook已经被集成在 iOS 6.0中。本教程的重点讲解如何利用苹果提供的类在iOS5.0和iOS6.0中部署Twitter和Facebook。

实例步骤

1. 创建一个简单View based application

2. 选择项目文件，然后选择"targets(目标)"，然后在 choose frameworks（选择框架）中添加Social.framework 和 Accounts.framework

3. 添加两个名为facebookPost 和 twitterPost的按钮，并为他们创建 ibActions。

4. 更新 ViewController.h 如下

```
#import <Social/Social.h>
#import <Accounts/Accounts.h>
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController

-(IBAction)twitterPost:(id)sender;
-(IBAction)facebookPost:(id)sender;

@end
```

5. 更新ViewController.m， 如下所示

```
#import "ViewController.h"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
}

- (void)didReceiveMemoryWarning
{
```

```
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

-(IBAction)facebookPost:(id)sender{

   SLComposeViewController *controller = [SLComposeViewController
   composeViewControllerForServiceType:SLServiceTypeFacebook];
   SLComposeViewControllerCompletionHandler myBlock =
   ^(SLComposeViewControllerResult result){
          if (result == SLComposeViewControllerResultCancelled)
          {
            NSLog(@"Cancelled");
          }
          else
          {
            NSLog(@"Done");
          }
          [controller dismissViewControllerAnimated:YES completion:
        };
   controller.completionHandler =myBlock;
   //Adding the Text to the facebook post value from iOS
   [controller setInitialText:@"My test post"];
   //Adding the URL to the facebook post value from iOS
   [controller addURL:[NSURL URLWithString:@"http://www.test.com"]]
   //Adding the Text to the facebook post value from iOS
   [self presentViewController:controller animated:YES completion:n
}

-(IBAction)twitterPost:(id)sender{
    SLComposeViewController *tweetSheet = [SLComposeViewController
    composeViewControllerForServiceType:SLServiceTypeTwitter];
    [tweetSheet setInitialText:@"My test tweet"];
    [self presentModalViewController:tweetSheet animated:YES];
}

@end
```

输出

运行该应用程序并单击 facebookPost 时我们将获得以下输出



当我们单击 twitterPost 时，我们将获得以下输出


Carrier
4:59 PM
Cancel
Tweet
Send
My test tweet
Add Location
127
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
.?123
space
@
#

IOS内存管理

简介

iOS下内存管理的基本思想就是引用计数，通过对象的引用计数来对内存对象的生命周期进行控制。具体到编程时间方面，主要有两种方式：

1：MRR（manual retain-release），人工引用计数，对象的生成、销毁、引用计数的变化都是由开发人员来完成。

2：ARC（Automatic Reference Counting），自动引用计数，只负责对象的生成，其他过程开发人员不再需要关心其销毁，使用方式类似于垃圾回收，但其实质还是引用计数。

面临的问题

根据苹果说明文档，面临的两个主要问题是：

释放或覆盖的数据仍然在使用。这将造成内存损坏，通常在应用程序崩溃，或者更糟，损坏用户数据。

不释放不再使用的数据会导致内存泄漏。分配的内存，内存泄漏不会释放，即使它从来没有再次使用。泄漏会导致应用程序的内存使用量日益增加，这反过来又可能会导致系统性能较差或死机。

内存管理规则

我们创建自己的对象，当他们不再需要的时候，释放他们。

保留需要使用的对象。如果没有必要必须释放这些对象。

不要释放我们没有拥有的对象。

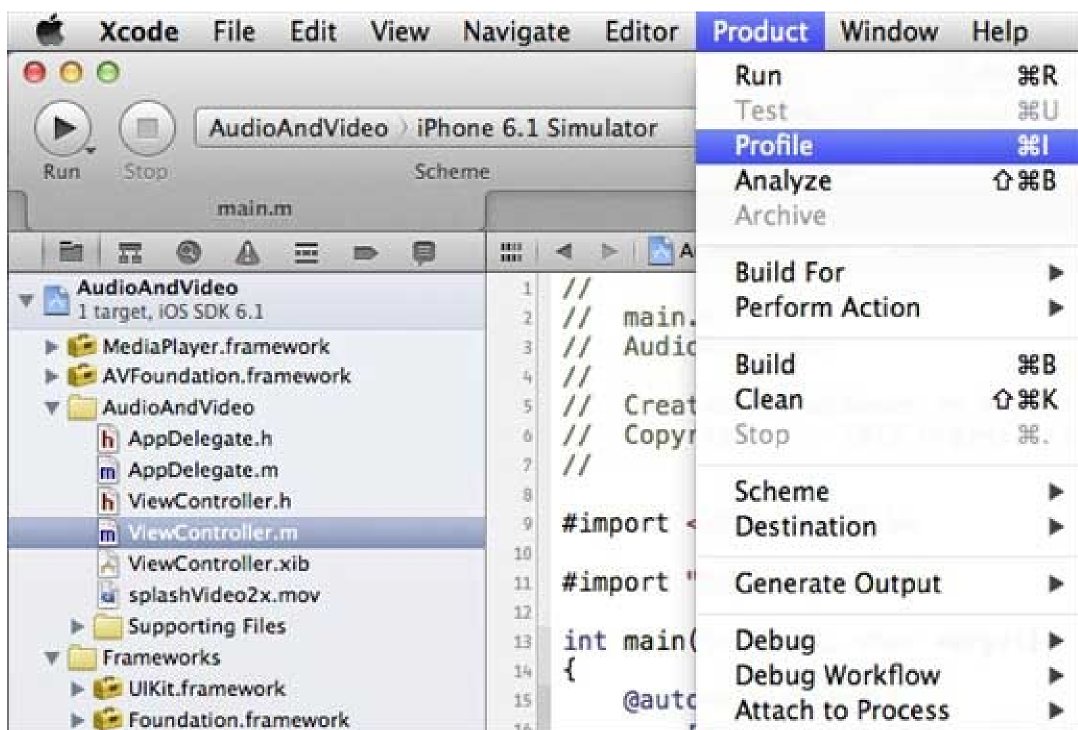
使用内存管理工具

可以用Xcode工具仪器的帮助下分析内存的使用情况。它包括的工具有活动监视器，分配，泄漏，僵尸等

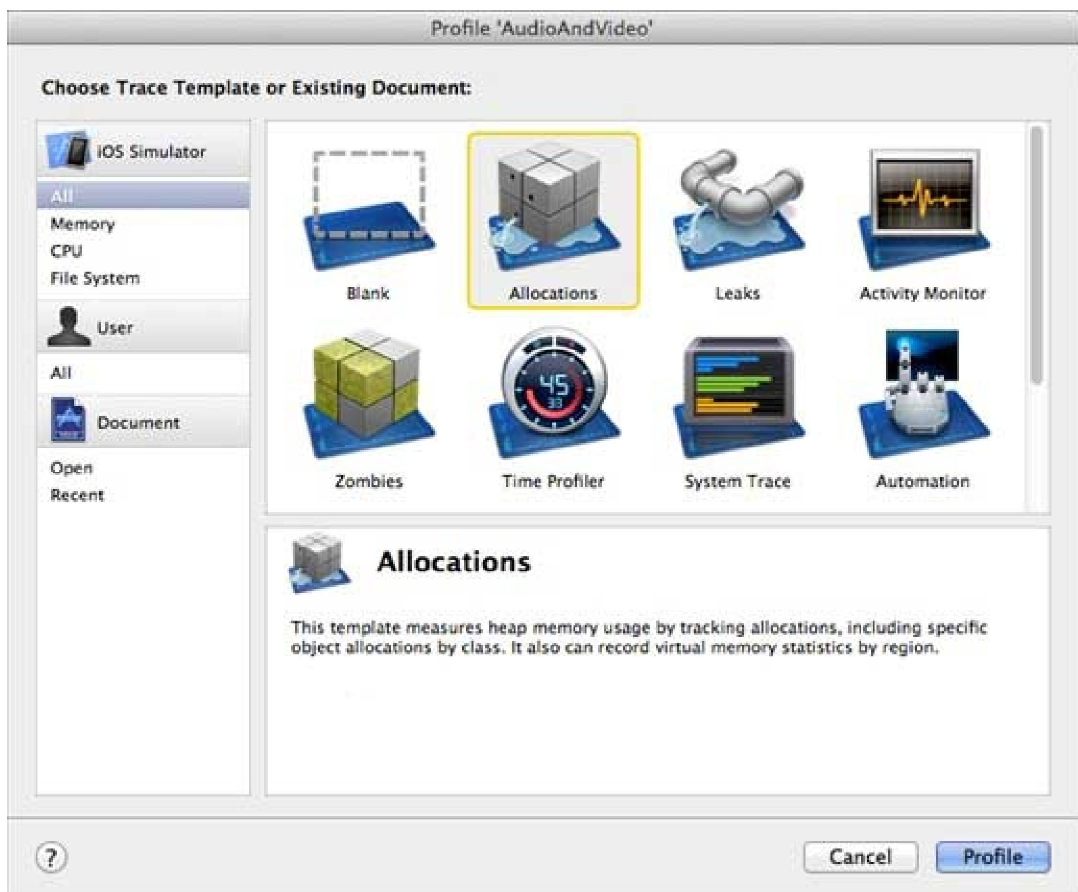
分析内存分配的步骤

1. 打开一个现有的应用程序。

2. 选择产品，配置文件如下所示

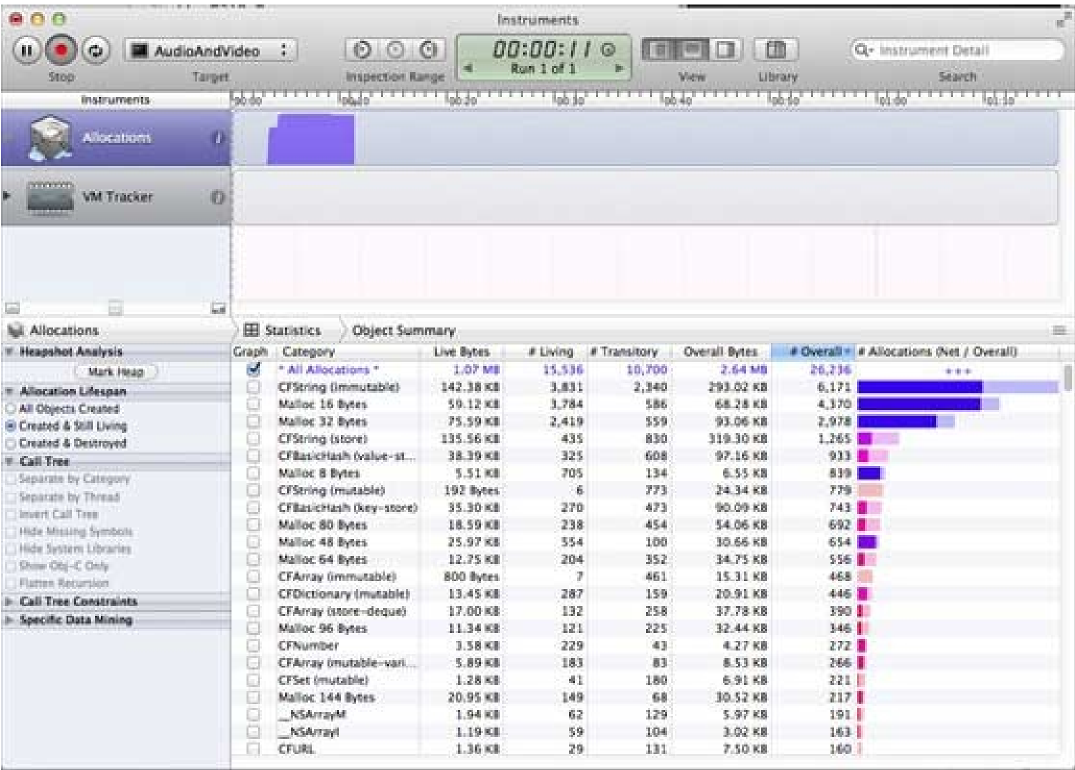


3.在以下界面中选择 Allocations 和 Profile。

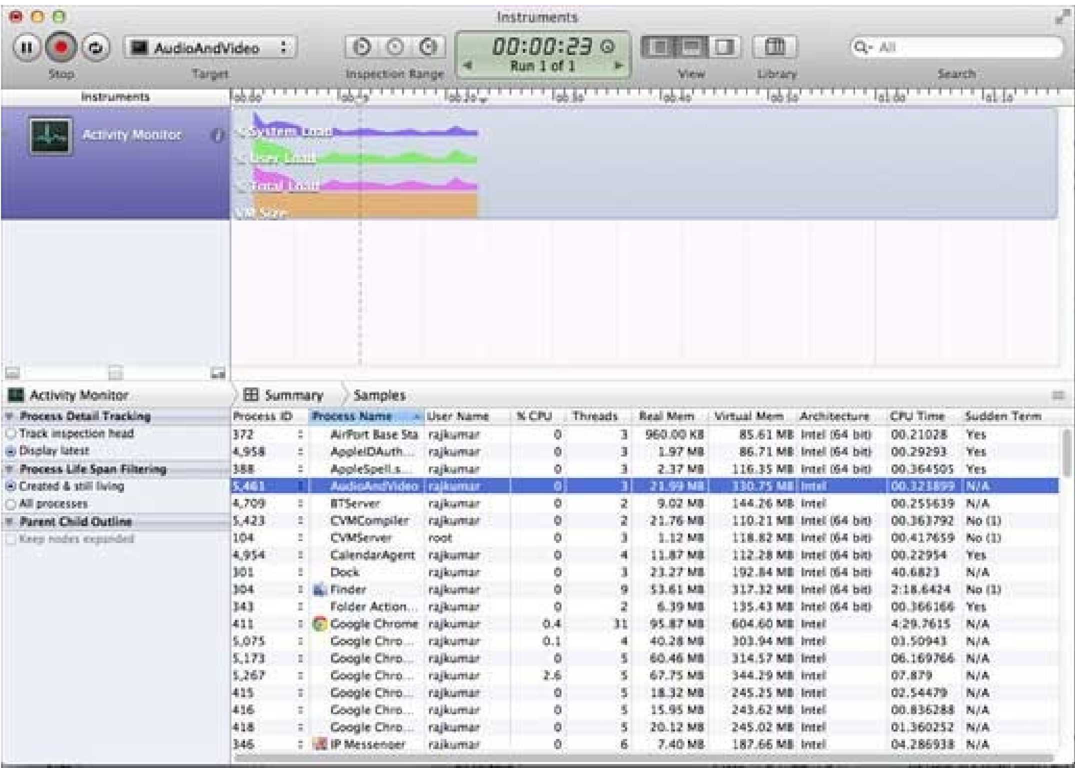


4. 我们可以看到不同对象的内存使用情况

5. 你可以切换视图控制器查看内存是否释放。



6.同样我们可以使用 Activity Monitor 来查看内存在应用程序中的分配的情况。



7. 这些工具可以帮助我们了解内存的使用情况及在什么地方可能发生泄漏。

IOS应用程序调试

简介

当我们做应用程序的时候，可能会犯各种错误，这可能会导致各种不同的错误。因此，为了修复这些错误或缺陷，我们需要来调试应用程序。

选择一个调试器

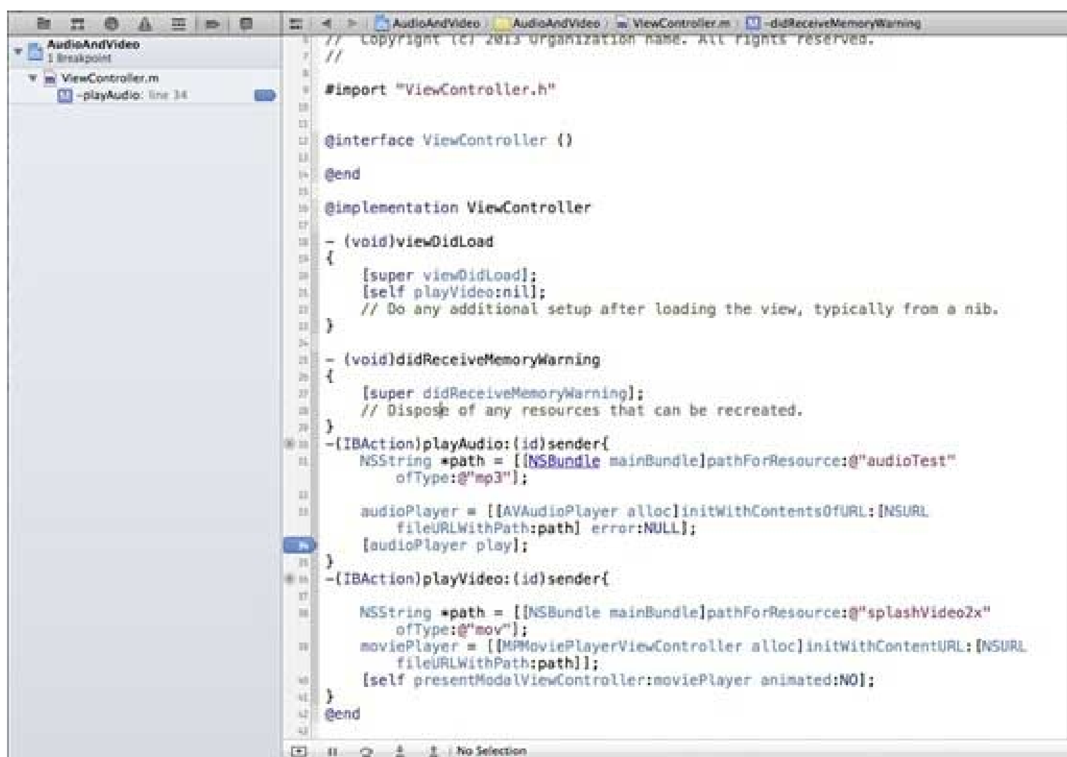
Xcode中调试器即 GDB 和 LLDB 调试器，GDB 是默认的。 LLDB是一个调试器是LLVM开源的编译器项目的一部分。您可以更改调试，编辑活动计划选项。

如何查找编码错误？

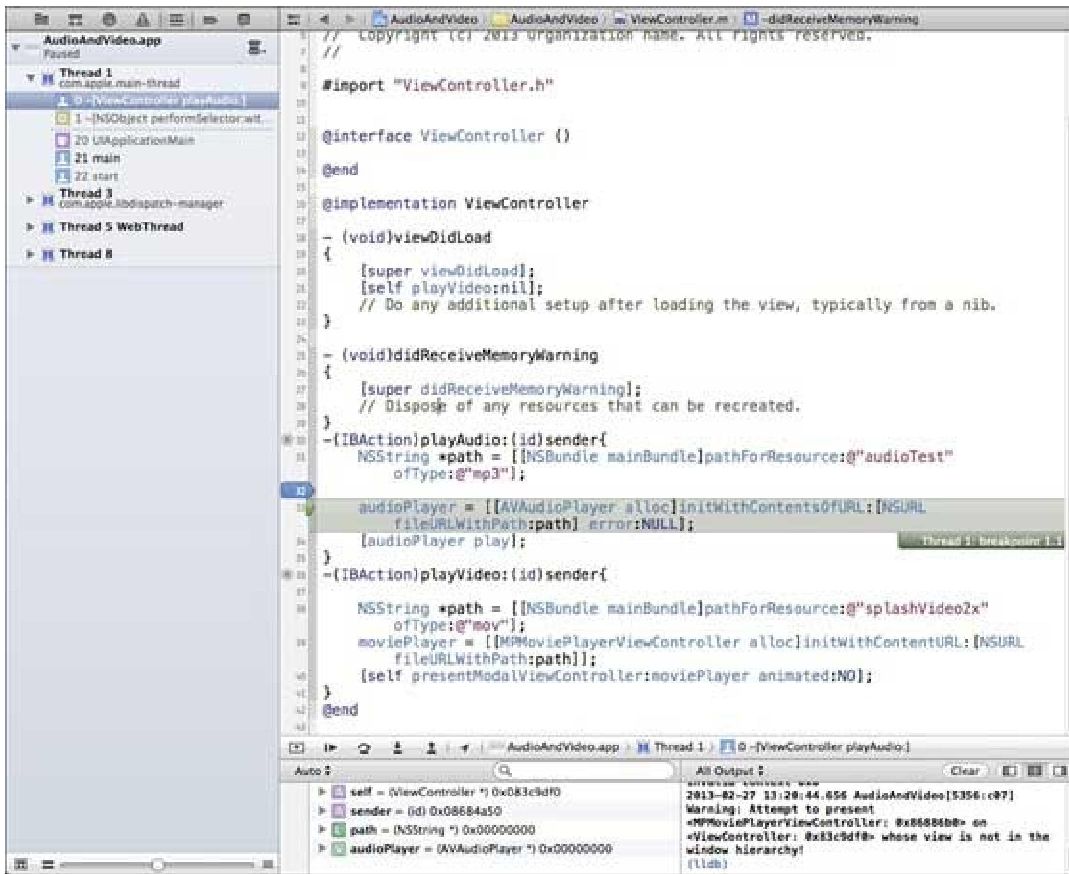
我们只需要建立我们的应用程序，代码被编译器编译，所有的消息，错误和警告将显示以及错误的原因，我们可以纠正他们。可以点击 product，然后点击"分析"，将在应用程序中可能发生的问题。

设置断点

断点帮助我们了解我们的应用程序对象，帮助我们找出许多缺陷，包括逻辑问题的不同状态。我们只需要点击创建一个断点的行号。我们可以通过点击并拖动它删除断点。如下所示



当我们运行应用程序并选择playVideo，按钮的应用程序将被暂停，我们来分析一下我们的应用程序的状态。当断点被触发时，我们将得到一个输出，如下图所示



可以轻松地确定哪个线程触发断点。在底部可以看到对象，如self，sender等，这些持有相应的对象的值，我们可以展开一些这些对象，看看他们每个的状态是什么。

要继续应用程序，我们在调试区选择继续按钮（最左边的按钮），如下图所示。其他选项包括步骤和单步跳过

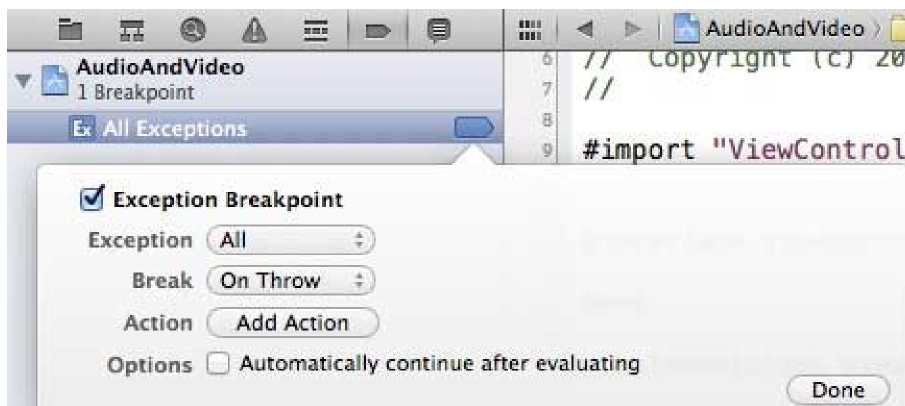


异常断点

我们也有异常断点，触发应用程序停止发生异常的位置。通过选择调试导航后选择"+"按钮，我们可以创建异常断点。将得到下面的窗口



然后，我们需要选择" Exception Breakpoint (添加异常)"断点，它会显示下面的窗口



下一步是什么？

你可以在 Xcode 4 用户指南 知道更多关于调试和其他Xcode功能的知识。

w3school jQuery Mobile 教程

来源：jQuery Mobile 教程

整理：飞龙

jQuery Mobile 简介

jQuery Mobile 是一个用于创建移动端web应用的的前端框架。

学习本教程前你需要先了解

在开始学习 jQuery Mobile 前, 你应该了解一下基础知识：

- HTML
- CSS
- jQuery

如果你想学习这些知识，你可以访问本站的首页。

什么是 jQuery Mobile?

jQuery Mobile 是针对触屏智能手机与平板电脑的网页开发框架。

jQuery Mobile 工作与所有主流的智能手机和平板电脑上：



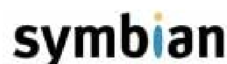














jQuery Mobile 构建于 jQuery 以及 jQuery UI类库之上，如果您了解 jQuery，您可以很容易的学习 jQuery Mobile。

jQuery Mobile 使用了极少的 HTML5、CSS3、JavaScript 和 AJAX 脚本代码来完成页面的布局渲染。

为什么使用 jQuery Mobile?

通过使用jQuery Mobile 可以 "写更少的代码，做更多的事情"：它可以通过一个灵活及简单的方式来布局网页，且兼容所有移动设备。

>  不同设备使用了不同开发语言，jQuery Mobile可以很好的兼容不容的设备或操作系统：

- Android 和 Blackberry（黑莓） 使用JAVA语言。
- iOS 使用 Objective C 语言
- Windows Phone 使用 C# 和 .net, 等。

> **jQuery Mobile** 解决了不同设备兼容的问题，因为它只使用**HTML**，**CSS**和**JavaScript**，这是所有移动网络浏览器的标准！

最好的阅读体验

尽管jQuery Mobile兼容所有的移动设备，但是并不能完全兼容PC机（由于有限的CSS3支持）。

为了更好的阅读本教程，建议您使用 **Google Chrome** 浏览器。

jQuery Mobile 安装

在你的网页中添加 jQuery Mobile

你可以通过以下几种方式将jQuery Mobile添加到你的网页中：

- 从 CDN 中加载 jQuery Mobile (推荐)
- 从jQuerymobile.com 下载 jQuery Mobile库

从 CDN 中加载 jQuery Mobile



> CDN的全称是Content Delivery Network，即内容分发网络。其基本思路是尽可能避开互联网上有可能影响数据传输速度和稳定性的瓶颈和环节，使内容传输的更快、更稳定。.

使用 jQuery 内核, 你不需要在电脑上安装任何东西; 你仅仅需要在你的网页中加载以下层叠样式 (.css) 和 JavaScript 库 (.js) 就能够使用 jQuery Mobile:

jQuery Mobile CDN:

```
<head>
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.3.2/jq
<script src="http://code.jquery.com/jquery-1.8.3.min.js"></script>
<script src="http://code.jquery.com/mobile/1.3.2/jquery.mobile-1.3.
</head>
```

下载 jQuery Mobile

如果你想将 jQuery Mobile 放于你的主机中,你可以从 jQuerymobile.com下载该文件。

```
<head>
<link rel=stylesheet href=jquery.mobile-1.3.2.css>
<script src=jquery.js></script>
<script src=jquery.mobile-1.3.2.js></script>
</head>
```

提示： 将下载的文件放置于与网页同一目录下。

>  你是否想知道为什么在 **<script>** 标签中 没有插入 **type="text/javascript" ?**
>
> 在 HTML5 已经不需要该属性。 JavaScript 在所有现代浏览器中是 HTML5 的默认脚本语言！

jQuery Mobile 页面

开始学习 jQuery Mobile

>  尽管jQuery Mobile兼容所有的移动设备，但是并不能完全兼容PC机（由于有限的CSS3支持）。
>
> 为了更好的阅读本教程，建议您使用 Google Chrome 浏览器。

实例

```
<body>
<div data-role="page">

  <div data-role="header">
    <h1>欢迎来到我的主页</h1>
  </div>

  <div data-role="content">
    <p>我现在是一个移动端开发者!!</p>
  </div>

  <div data-role="footer">
    <h1>底部文本</h1>
  </div>

</div>
</body>
```

实例解析：

- data-role="page" 是在浏览器中显示的页面。
- data-role="header" 是在页面顶部创建的工具条 (通常用于标题或者搜索按钮)
- data-role="content" 定义了页面的内容，比如文本， 图片，表单，按钮等。
- data-role="footer" 用于创建页面底部工具条。
- 在这些容器中你可以添加任何 HTML 元素 - 段落, 图片, 标题, 列表等。

>  jQuery Mobile 依赖 HTML5 data-* 属性来支持各种 UI 元素、过渡和页面结构。不支持它们的浏览器将以静默方式弃用它们。

在页面中添加 jQuery Mobile

使用 jQuery Mobile, 你可以再单个 HTML 文件中创建多个不同的页面。

你可以使用不同的href属性来区分使用同一个唯一id的页面：

实例

```
<div data-role="page" id="pageone">
  <div data-role="content">
    <a href="#pagetwo">Go to Page Two</a>
  </div>
</div>

<div data-role="page" id="pagetwo">
  <div data-role="content">
    <a href="#pageone">Go to Page One</a>
  </div>
</div>
```

注意： 当web应用有大量的内容（文本，图片，脚本等）将会严重影响加载时间。如果你不想使用内页链接可以使用外部文件：

```
<a href="externalfile.html">访问外部文件</a>
```

页面作为对话框使用

对话框是用于显示页面信息显示或者表单信息的输入。

在链接中添加data-rel="dialog"让用户点击链接时弹出对话框:

实例

```
<div data-role="page" id="pageone">
  <div data-role="content">
    <a href="#pagetwo" data-rel="dialog">Go to Page Two</a>
  </div>
</div>

<div data-role="page" id="pagetwo">
  <div data-role="content">
    <a href="#pageone">Go to Page One</a>
  </div>
</div>
```

jQuery Mobile 页面切换

jQuery Mobile 包含 CSS3 效果让您选择页面打开的方式。

jQuery Mobile 页面切换效果

jQuery Mobile 提供了各种页面切换到下一个页面的效果。

注意：为了实现页面切换效果，浏览器必须支持 CSS3 3D 切换：

- Internet Explorer 10 支持 3D 切换（早期版本不支持）
- Opera 不支持 3D 切换

页面切换效果可被应用于任何使用 data-transition 属性的链接或表单提交：

```
<a href="#anylink" data-transition="slide">切换到第二个页面</a>
```

下面的表格显示了通过使用 data-transition 属性后可用的页面切换：

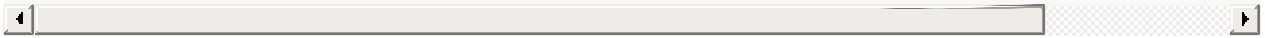
页面切换	描述	尝试一下
fade	默认。淡入到下一页	
flip	从后向前翻转到下一页	
flow	抛出当前页，进入下一页	
pop	像弹出窗口一样进入下一页	
slide	从右到左滑动到下一页	
slidefade	从右到左滑动并淡入到下一页	
slideup	从下到上滑动到下一页	
slidedown	从上到下滑动到下一页	
turn	翻到下一页	
none	没有切换效果	

 在 jQuery Mobile 的所有链接上，默认使用淡入淡出的效果（如果浏览器支持）。

提示：上面的所有效果支持后退行为。例如，如果您想要页面从左向右滑动，而不是从右向左滑动，请使用带有 "reverse" 值的 data-direction 属性。在后退按钮上这是默认的。

实例

```
<a href="#pagetwo" data-transition="slide" data-direction="reverse"
```



jQuery Mobile 按钮

Mobile 应用程序是建立在您想要显示的简单的点击事物上。



在 jQuery Mobile 中创建按钮

在 jQuery Mobile 中，按钮可通过三种方式创建：

- 使用 <button> 元素
- 使用 <input> 元素
- 使用带有 data-role="button" 的 <a> 元素

<button>

```
<button>按钮</button>
```

<input>

```
<input type="button" value="按钮">
```

<a>

```
<a href="#" data-role="button">按钮</a>
```

>  在 jQuery Mobile 中，按钮会自动样式化，让它们在移动设备上更具吸引力和可用性。我们推荐您使用带有 **data-role="button"** 的 **<a>** 元素在页面间进行链接，使用 **<input>** 或 **<button>** 元素进行表单提交。

导航按钮

如需通过按钮在页面间进行链接，请使用带有 data-role="button" 属性的 <a> 元素：

实例

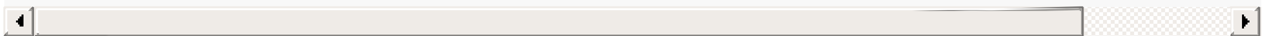
```
<a href="#pagetwo" data-role="button">访问第二个页面</a>
```

内联按钮

默认情况下，按钮占满整个屏幕宽度。如果你想要一个仅是与内容一样宽的按钮，或者如果您想要并排显示两个或多个按钮，请添加 data-inline="true"：

实例

```
<a href="#pagetwo" data-role="button" data-inline="true">访问第二个
```



组合按钮

jQuery Mobile 提供了一个简单的方法来将按钮组合在一起。

请把 data-role="controlgroup" 属性和 data-type="horizontal|vertical" 一起使用来规定是否水平或垂直组合按钮：

实例

```
<div data-role="controlgroup" data-type="horizontal">
<a href="#anylink" data-role="button">按钮 1</a>
<a href="#anylink" data-role="button">按钮 2</a>
<a href="#anylink" data-role="button">按钮 3</a>
</div>
```

 默认情况下，组合按钮是垂直组合，它们之间没有外边距和空间。并且只有第一个和最后一个按钮是圆角，以便它们组合在一起的时候创建一个漂亮的外观。

后退按钮

如需创建后退按钮，请使用 data-rel="back" 属性（这会忽略锚的 href 值）：

实例

```
<a href="#" data-role="button" data-rel="back">返回</a>
```

更多用于按钮的 **data-*** 属性

属性	值	描述	实例
data-corners	true | false	规定按钮是否圆角	
data-mini	true | false	规定按钮是否更小	
data-shadow	true | false	规定按钮是否有阴影	

如需查看所有 jQuery Mobile data-* 属性的完整参考手册，请访问我们的 jQuery Mobile Data 属性参考手册。

下一章演示如何附加图标到您的按钮上。

jQuery Mobile 按钮图标

jQuery Mobile 提供了一套让按钮看起来更称心如意的图标。



添加图标到 jQuery Mobile 按钮

如需添加图标到您的按钮，请使用 data-icon 属性：

```
<a href="#anylink" data-role="button" data-icon="search">Search</a>
```

提示：您也可以在 <button> 或 <input> 元素上使用 data-icon 属性。

下面我们列出一些 jQuery Mobile 提供的可用图标：

属性值	描述	图标	实例
data-icon="arrow-l"	左箭头		
data-icon="arrow-r"	右箭头		
data-icon="delete"	删除		
data-icon="info"	信息		
data-icon="home"	首页		
data-icon="back"	后退		
data-icon="search"	搜索		
data-icon="grid"	网格		

如需查看所有 jQuery Mobile 按钮图标的完整参考手册，请访问我们的 jQuery Mobile 图标参考手册。

定位图标

您也可以规定图标定位在按钮的什么部位：顶部（top）、右侧（right）、底部（bottom）、左侧（left）。

请使用 data-iconpos 属性来指定位置：

图标的位置：

```
<a href="#link" data-role="button" data-icon="search" data-iconpos=
<a href="#link" data-role="button" data-icon="search" data-iconpos=
<a href="#link" data-role="button" data-icon="search" data-iconpos=
<a href="#link" data-role="button" data-icon="search" data-iconpos=
```

 默认情况下，所有的按钮图标被放置到左侧。

只显示图标

如果只想显示图标，请设置 data-iconpos 为 "notext"：

后退：

```
<a href="#link" data-role="button" data-icon="search" data-iconpos=
```

jQuery Mobile 工具栏

jQuery Mobile 工具栏

工具栏元素通常位于头部和尾部内 - 让导航易于访问：

这是一个简单的弹窗！
阅读教程了解如何使用弹窗！

头部栏

头部栏一般包含页面标题/logo 或一两个按钮（通常是首页、选项或搜索）。

您可以添加按钮到头部的左侧或右侧。

下面的代码，将添加一个按钮到头部标题文本的左侧，添加一个按钮到头部标题文本的右侧：

实例

```
<div data-role="header">
<a href="#" data-role="button">首页</a>
<h1>欢迎来到我的主页</h1>
<a href="#" data-role="button">搜索</a>
</div>
```

下面的代码，将添加一个按钮到头部标题文本的左侧：

```
<div data-role="header">
<a href="#" data-role="button">首页</a>
<h1>欢迎来到我的主页</h1>
</div>
```

但是，如果您把按钮链接放置在 <h1> 元素之后，将无法显示右侧的文本。要添加一个按钮到头部标题的右侧，请指定 class 为 "ui-btn-right"：

实例

```
<div data-role="header">
<h1>欢迎来到我的主页</h1>
<a href="#" data-role="button" class="ui-btn-right">搜索</a>
</div>
```

 头部可以包含一个或两个按钮，而尾部没有限制。

尾部栏

尾部栏比头部栏更灵活 - 在整个页面中它们更具功能性和可变性，因此可以包含尽可能多的按钮：

实例

```
<div data-role="footer">
<a href="#" data-role="button">在 Facebook上关注我</a>
<a href="#" data-role="button">在Twitter上关注我</a>
<a href="#" data-role="button">在Instagram上关注我</a>
</div>
```

注意：尾部的样式与头部不同（没有内边距和空间，且按钮不居中）。为了解决这个问题，请把 "ui-btn" 放置在尾部的 class 上：

实例

```
<div data-role="footer" class="ui-btn">
```

您还可以将尾部中的按钮进行水平或垂直组合：

实例

```
<div data-role="footer" class="ui-btn">
<div data-role="controlgroup" data-type="horizontal">
<a href="#" data-role="button" data-icon="plus">在Facebook上关注我</
<a href="#" data-role="button" data-icon="plus">在Twitter上关注我</a
<a href="#" data-role="button" data-icon="plus">在Instagram上关注我<
</div>
</div>
```

定位头部栏和尾部栏

头部和尾部可以通过三种方式进行定位：

- Inline - 默认。头部栏和尾部栏与页面内容内联。
- Fixed - 头部栏和尾部栏固定在页面的顶部和底部。
- Fullscreen - 与 Fixed 定位模式基本相同，头部栏和尾部栏固定在页面的顶部和底部。但是当他工具栏滚动出屏幕之外时，不会自动重新显示，除非点击屏幕，这对于图片或视频类有提升代入感的应用是非常有用的。注意这种模式下工具栏会遮住页面内容，所以最好用在比较特殊的场合下。

使用 data-position 属性来定位头部栏和尾部栏：

Inline 定位（默认）

```
<div data-role="header" data-position="inline"></div>
<div data-role="footer" data-position="inline"></div>
```

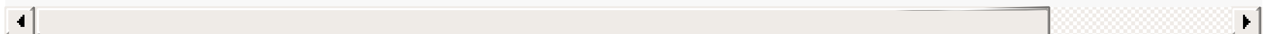
Fixed 定位

```
<div data-role="header" data-position="fixed"></div>
<div data-role="footer" data-position="fixed"></div>
```

要启用全屏定位，请使用 data-position="fixed"，并添加 data-fullscreen 属性到元素：

Fullscreen 定位

```
<div data-role="header" data-position="fixed" data-fullscreen="true
<div data-role="footer" data-position="fixed" data-fullscreen="true
```



提示：全屏定位适用于照片、图像和视频。

提示：固定定位和全屏定位中，通过点击屏幕将隐藏和显示头部栏和尾部栏。

jQuery Mobile 导航栏

jQuery Mobile 导航栏

导航栏是由一组水平排列的链接组成，通常包含在头部或尾部内。

默认情况下，导航栏中的链接将自动变成按钮（不需要 data-role="button"）。

使用 data-role="navbar" 属性来定义导航栏：

实例

```
<div data-role="header">
<div data-role="navbar">
<ul>
<li><a href="#anylink">首页</a></li>
<li><a href="#anylink">页面二</a></li>
<li><a href="#anylink">搜索</a></li>
</ul>
</div>
</div>
```

>  默认情况下，按钮的宽度与它的内容一样。使用一个无序列表来平均地划分按钮的宽度：1 个按钮占 100% 宽度，2 个按钮则各占 50% 的宽度，3 个按钮则每个占 33,3% 的宽度，依此类推。然而，如果您在导航栏中指定了超过 5 个按钮，将会拆成多行（查看"更多实例"）。

激活按钮

当导航栏中的某个链接被点击，它将获得被选中（按下）的外观。

如果想在不点击链接时获得这种外观，请使用 class="ui-btn-active"：

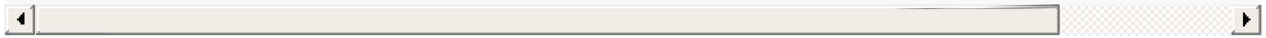
实例

```
<li><a href="#anylink" class="ui-btn-active">首页</a></li>
```

对于多个页面，您可能想要每个按钮的选中外观代表当前用户所在的页面。要做到这一点，请添加 "ui-state-persist" 和 "ui-btn-active" 到链接的 class：

实例

```
<li><a href="#anylink" class="ui-btn-active ui-state-persist">首页<
```



更多实例

内容中的导航栏 如何在 data-role="content" 内添加导航栏。

尾部中的导航栏 如何在尾部内添加导航栏。

导航栏中的定位图标 如何在尾部内的导航栏中定位图标。

超过 5 个按钮 导航栏中 10 个按钮的演示。

jQuery Mobile 可折叠块

可折叠内容块

可折叠块允许您隐藏或显示内容 - 对于存储部分信息很有用。

如需创建一个可折叠的内容块，需要为容器添加 data-role="collapsible" 属性。在容器（div）内，添加一个标题元素（H1-H6），后跟您想要进行扩展的 HTML 标记：

实例

```
<div data-role="collapsible">
<h1>点击我 - 我可以折叠!</h1>
<p>我是可折叠的内容。</p>
</div>
```

默认情况下，内容是被折叠起来的。如需在页面加载时展开内容，请使用 data-collapsed="false"：

实例

```
<div data-role="collapsible" data-collapsed="false">
<h1>点击我 - 我可以折叠!</h1>
<p>I'm 现在我默认是展开的。</p>
</div>
```

嵌套可折叠块

可折叠的内容块是可以彼此嵌套的：

实例

```
<div data-role="collapsible">
<h1>点击我 - 我可以折叠!</h1>
<p>我是被展开的内容。</p>
<div data-role="collapsible">
<h1>点击我 - 我是嵌套的可折叠块!</h1>
<p>我是嵌套的可折叠块中被展开的内容。</p>
</div>
</div>
```

 可折叠的内容块可以根据您的需要进行多次嵌套。

可折叠集合

可折叠集合是将可折叠块组合在一起（就像手风琴一样）。当一个新的块被展开时，所有其他的块都会被折叠起来。

创建若干个可折叠的内容块，然后把可折叠内容块用带有 data-role="collapsible-set" 的新容器包围起来：

实例

```
<div data-role="collapsible-set">
<div data-role="collapsible">
<h1>点击我 - 我可以折叠！</h1>
<p>我是被展开的内容。</p>
</div>
<div data-role="collapsible">
<h1>点击我 - 我可以折叠!</h1>
<p>我是被展开的内容。</p>
</div>
</div>
```

更多实例

通过 data-inset 属性取消圆角 如何取消可折叠块上的圆角。

通过 data-mini 属性迷你化可折叠块 如何让可折叠块更小。

通过 data-collapsed-icon 和 data-expanded-icon 改变图标 如何改变可折叠块的图标（默认是 + 和 -）。

jQuery Mobile 网格

jQuery Mobile 布局网格

jQuery Mobile 提供了一套基于 CSS 的分列布局。然而，在移动设备上，由于考虑手机的屏幕宽度狭窄，一般不建议使用分栏分列布局。

但有时您想要将较小的元素（如按钮或导航标签）并排地排列在一起，就像是在一个表格中一样。这种情况下，推荐使用分列布局。

网格中的列是等宽的（合计是 100%），没有边框、背景、margin 或 padding。

这里有四种布局网格可供使用：

网格类	列	列宽	对应	实例
ui-grid-a	2	50% / 50%	ui-block-a|b	
ui-grid-b	3	33% / 33% / 33%	ui-block-a|b|c	
ui-grid-c	4	25% / 25% / 25% / 25%	ui-block-a|b|c|d	
ui-grid-d	5	20% / 20% / 20% / 20% / 20%	ui-block-a|b|c|d|e	

 在列容器内，子元素拥有的 class 为 ui-block-a|b|c|d|e 取决于列数。列会浮动并排。

实例 **1**:class 为 ui-grid-a（两列布局），您必须指定 ui-block-a 和 ui-block-b 的两个子元素。

实例 **2**:class 为 ui-grid-b（三列布局），则必须添加 ui-block-a、ui-block-b 和 ui-block-c 的三个子元素。

自定义网格

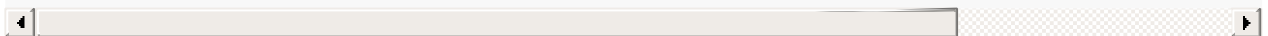
通过使用CSS，您可以自定义列块：

实例

```
<style>
.ui-block-a,
.ui-block-b,
.ui-block-c
{
background-color: lightgray;
border: 1px solid black;
height: 100px;
font-weight: bold;
text-align: center;
padding: 30px;
}
</style>
```

您也可以通过使用内嵌样式来自定义块：

```
<div class="ui-block-a" style="border: 1px solid black;"><span>Text
```



多行

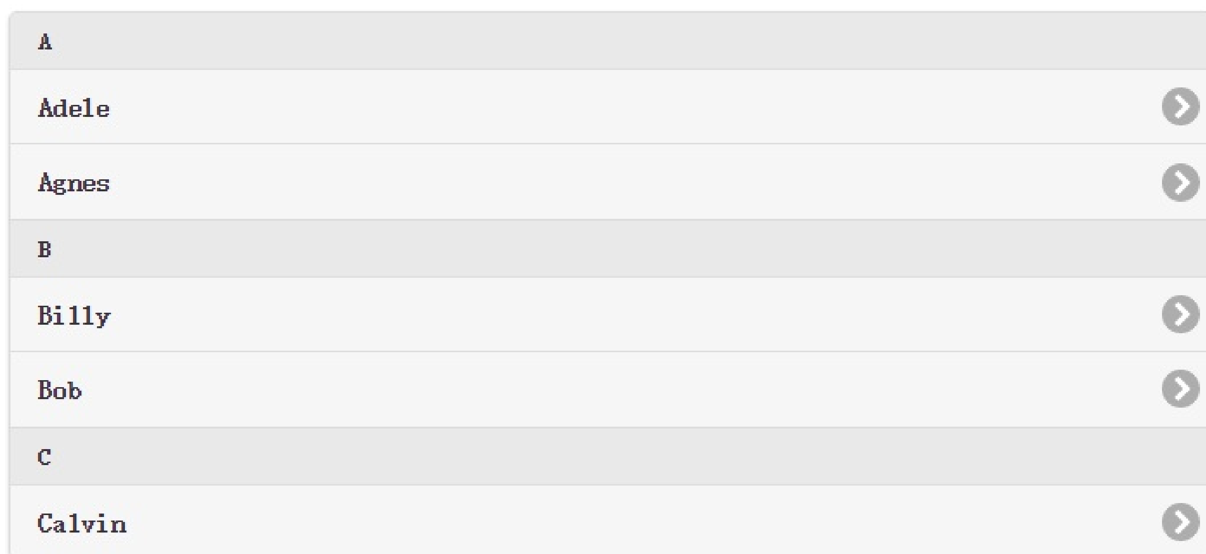
在列中也可以有多个行。

注意：ui-block-a-class 总是创建一个新行：

实例

```
<div class="ui-grid-b">
<div class="ui-block-a"><span>Some Text</span></div>
<div class="ui-block-b"><span>Some Text</span></div>
<div class="ui-block-c"><span>Some Text</span></div>
<div class="ui-block-a"><span>Some Text</span></div>
<div class="ui-block-b"><span>Some Text</span></div>
<div class="ui-block-a"><span>Some Text</span></div>
</div>
```

jQuery Mobile 列表视图



jQuery Mobile 列表视图

jQuery Mobile 中的列表视图是标准的HTML 列表; 有序(<ol>) 和 无序(<ul>).

列表视图是jQuery Mobile中功能强大的一个特性。它会使标准的无序或有序列表应用更广泛。应用方法就是在ul或ol标签中添加data-role="listview"属性。在每个项目(<li>)中添加链接，用户可以点击它：

实例

```
<ol data-role="listview">
  <li><a href="#">列表项m</a></li>
</ol>

<ul data-role="listview">
  <li><a href="#">列表项</a></li>
</ul>
```

列表样式的圆角和边缘，使用 data-inset="true" 属性设置:

实例

```
<ul data-role="listview" data-inset="true">
```

 默认情况下，列表项的链接会自动变成一个按钮 (不需要 data-role="button")。 |

列表分隔

列表项也可以转化为列表分割项，用来组织列表，使列表项成组。

指定列表分割，给列表项<li>元素添加 data-role="list-divider" 属性即可：

实例

```
<ul data-role="listview">
 <li data-role="list-divider">欧洲</li>
  <li><a href="#">法国</a></li>
  <li><a href="#">德国</a></li>
</ul>
```

如果你有一个字母顺序排列的列表，（例如一个电话簿）通过 <ol> 或者 元素的 data-autodividers="true" 属性设置可以配置为自动生成的项目的分隔:

实例

```
<ul data-role="listview" data-autodividers="true">
  <li><a href="#">Adele</a></li>
  <li><a href="#">Agnes</a></li>
  <li><a href="#">Billy</a></li>
  <li><a href="#">Calvin</a></li>
  ...
</ul>
```

 data-autodividers="true" 可以配置为自动生成的项目的分隔。默认情况下，创建的分隔文本是列表项文本的第一个大写字母。

搜索过滤

jquery Mobile提供一个非常简单的方法，实现客户端搜索功能，筛选列表的选项。只需添加 data-filter="true" 属性即可:

实例

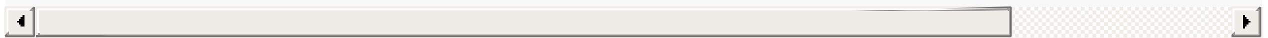
```
<ul data-role="listview" data-filter="true"></ul>
```

默认情况下，搜索输入框默认的字符为 "Filter items..."。

通过设置mobileinit事件的绑定程序或者给 $.mobile.listview.prototype.options.filterPlaceholder 选项设置一个字符串，或者给列表设置 data-filter-placeholder 属性，可以设置搜索输入框的默认字符:

实例

```
<ul data-role="listview" data-filter="true" data-filter-placeholde
```



更多实例

只读列表 如何创建一个不带链接的列表 (不会是个按钮且不可点击)。

jQuery Mobile 列表内容

浏览器

Google Chrome

Google Chrome 是免费的开源 web 浏览器。发布于 2008 年。

Mozilla Firefox

Firefox 是来自 Mozilla 的 web 浏览器。发布于 2004 年。

jQuery Mobile 列表缩略图

大于 16x16px 的图像，请在链接中添加 <img> 元素。

jQuery Mobile 将自动缩放图像到 80x80px:

实例

```
<ul data-role="listview">
  <li><a href="#"><img src="chrome.png"></a></li>
</ul>
```

使用标准的HTML添加列表信息：

实例

```
<ul data-role="listview">
  <li>
    <a href="#">
    <img src="chrome.png">
    <h2>Google Chrome</h2>
    <p>Google Chrome 免费的开源 web 浏览器。发布于 2008 年。</p>
    </a>
  </li>
</ul>
```

jQuery Mobile 列表图标

在列表 <img> 元素使用 class="ui-li-icon" 添加 16x16px 图标：

实例

```
<li><a href="#"><img src="us.png" alt="USA" class="ui-li-icon">USA<
```

分割按钮

在JQuery Mobile的列表中，有时需要对选项内容做两个不同的操作，这时，需要对选项中的链接按钮进行分割。实现分割的方法是在<li>元素中再增加一个<a>元素，便可以在页面实现分割效果。

jQuery Mobile 会自动设置第二个链接为蓝色箭头的图标，图标的链接文字（如果有的话）将在用户将鼠标悬停在 图标时显示:

实例

```
<ul data-role="listview">
  <li>
    <a href="#"><img src="chrome.png"></a>
    <a href="#">Some Text</a>
  </li>
</ul>
```

添加一些页面和对话框使链接功能更加丰富：

实例

```
<ul data-role="listview">
  <li>
    <a href="#"><img src="chrome.png"></a>
    <a href="#download" data-rel="dialog">下载浏览器</a>
  </li>
</ul>
```

气泡数字

气泡数字是用来显示列表项相关的数字，如在一个邮箱的邮件：

如需添加气泡数字，请使用行内元素，比如 <span>，设置 class "ui-li-count" 属性并添加数字：

实例

```
<ul data-role="listview">
  <li><a href="#">收件箱<span class="ui-li-count">25</span></a></li
  <li><a href="#">发件箱<span class="ui-li-count">432</span></a></l
  <li><a href="#">垃圾箱<span class="ui-li-count">7</span></a></li>
</ul>
```

注意：显示一个正确的气泡数字，必须修改编程方式。 这将在以后的章节解释。

更多实例

改变列表项的默认链接图标 如何设置列表项的默认链接图标(默认是右箭头).

可折叠的列表 如何创建显示/隐藏的列表。

更多内容格式 如何制作一个日历。

jQuery Mobile 表单

jQuery Mobile 会自动为 HTML 表单自动添加样式，让它们看起来更具吸引力，触摸起来更具友好性。

jQuery Mobile 表单

全名： 你的姓名是？

需要查找什么？ 搜索内容

今天的日期： 年 /月/日

选择喜爱的颜色： 红色

切换开关： Off

选择喜欢的电影：

蜘蛛侠

变形金刚

碟中谍

jQuery Mobile 表单结构

jQuery Mobile 使用 CSS 为 HTML 表单元素添加样式，让它们更具吸引力，更易于使用。

在 jQuery Mobile 中，您可以使用下列表单控件：

- 文本输入框
- 搜索输入框
- 单选按钮
- 复选框
- 选择菜单
- 滑动条
- 翻转拨动开关

当使用 jQuery Mobile 表单时，您应当知道：

- <form> 元素必须有一个 method 和一个 action 属性
- 每个表单元素必须有一个唯一的 "id" 属性。id 必须是整个站点所有页面上唯一的。这是因为 jQuery Mobile 的单页导航机制使得多个不同页面在同一时间被呈现
- 每个表单元素必须有一个标签。设置标签的 **for** 属性来匹配元素的 id

实例

```
<form method="post" action="demoform.html">
<label for="fname">姓名:</label>
<input type="text" name="fname" id="fname">
</form>
```

如需隐藏标签，请使用 class ui-hidden-accessible。这在您把元素的 placeholder 属性作为标签时经常用到：

实例

```
<form method="post" action="demoform.html">
<label for="fname" class="ui-hidden-accessible">姓名:</label>
<input type="text" name="fname" id="fname" placeholder="姓名...">
</form>
```

Field 容器

如需让标签和表单元素看起来更适应宽屏，请用带有 data-role="fieldcontain" 属性的 <div> 或 <fieldset> 元素包围 label/form 元素：

实例

```
<form method="post" action="demoform.html">
<div data-role="fieldcontain">
<label for="fname">姓:</label>
<input type="text" name="fname" id="fname">
<label for="lname">名:</label>
<input type="text" name="lname" id="lname">
</div>>
</form>
```

>  fieldcontain 属性基于页面的宽度为标签和表单控件添加样式。当页面的宽度大于 480px 时，它会自动把标签放置在与表单控件同一线上。当页面的宽度小于 480px 时，标签会被放置在表单元素的上面。

提示：为了防止 jQuery Mobile 为可点击元素自动添加样式，请使用 data-role="none" 属性：

实例

```
<label for="fname">姓名:</label>
<input type="text" name="fname" id="fname" data-role="none">
```

>  **jQuery Mobile** 中的表单提交
>
> jQuery Mobile 通过 AJAX 自动处理表单提交，并将试图集成服务器响应到应用程序的 DOM 中。

jQuery Mobile 表单输入元素

jQuery Mobile 文本输入框

输入字段是通过标准的 HTML 元素编码的，jQuery Mobile 将为它们添加样式使其看起来更具吸引力，在移动设备上更易使用。您也能使用新的 HTML5 的 <input> 类型：

实例

```
<form method="post" action="demo_form.php">
<div data-role="fieldcontain">
<label for="fullname">全名:</label>
<input type="text" name="fullname" id="fullname">

<label for="bday">生日:</label>
<input type="date" name="bday" id="bday">

<label for="email">E-mail:</label>
<input type="email" name="email" id="email" placeholder="你的电子邮箱
</div>
</form>
```

提示：请使用 placeholder 来指定一个简短的描述，用来描述输入字段的期望值：

```
<input placeholder="_sometext_">
```

文本域

对于多行文本输入可使用 <textarea> 。

注意：当您键入一些文本时，文本域会自动调整大小以适应新增加的行。

实例

```
<form method="post" action="demo_form.php">
<div data-role="fieldcontain">
<label for="info">附加信息:</label>
<textarea name="addinfo" id="info"></textarea>
</div>
</form>
```

搜索输入框

type="search" 类型的输入框是在 HTML5 中新增的，它是为输入搜索定义文本字段：

实例

```
<form method="post" action="demo_form.php">
<div data-role="fieldcontain">
<label for="search">搜索:</label>
<input type="search" name="search" id="search">
</div>
</form>
```

单选按钮

当用户在有限数量的选择中仅选取一个选项时，使用单选按钮。

为了创建一系列单选按钮，请添加带有 type="radio" 的 input 以及相应的 label。把单选按钮包围在 <fieldset> 元素内。您也可以添加一个 <legend> 元素来定义 <fieldset> 的标题。

提示：请使用 data-role="controlgroup" 来把按钮组合在一起：

实例

```
<form method="post" action="demo_form.php">
<fieldset data-role="controlgroup">
<legend>Choose your gender:</legend>
<label for="male">Male</label>
<input type="radio" name="gender" id="male" value="male">
<label for="female">Female</label>
<input type="radio" name="gender" id="female" value="female">
</fieldset>
</form>
```

复选框

当用户在有限数量的选择中选取一个或多个选项时，使用复选框：

实例

```
<form method="post" action="demo_form.php">
<fieldset data-role="controlgroup">
<legend>Choose as many favorite colors as you'd like:</legend>
<label for="red">Red</label>
<input type="checkbox" name="favcolor" id="red" value="red">
<label for="green">Green</label>
<input type="checkbox" name="favcolor" id="green" value="green">
<label for="blue">Blue</label>
<input type="checkbox" name="favcolor" id="blue" value="blue">
</fieldset>
</form>
```

更多实例

如需水平组合单选按钮或复选框，请使用 data-type="horizontal"：

实例

```
<fieldset data-role="controlgroup" data-type="horizontal">
```

您也可以用一个 field 容器包围 <fieldset>：

实例

```
<div data-role="fieldcontain">
<fieldset data-role="controlgroup">
<legend>请选择您的性别:</legend>
</fieldset>
</div>
```

如果您想要您的按钮中的一个预先选中，请使用 HTML 中 <input> 的 checked 属性：

实例

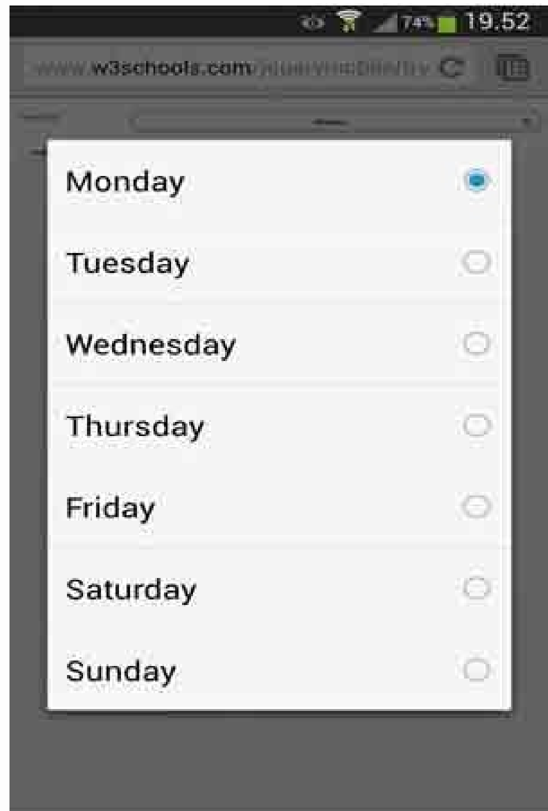
```
<input type="radio" checked>
<input type="checkbox" checked>
```

jQuery Mobile 表单选择菜单

jQuery Mobile 选择菜单

Iphone 上的选择菜单：Android/SGS4 设备上的选择菜单：





<select> 元素创建带有若干选项的下拉列表。

<select> 元素内的 <option> 元素定义了列表中的可用选项：

实例

```
<form method="post" action="demoform.html">
<fieldset data-role="fieldcontain">
<label for="day">Select Day</label>
<select name="day" id="day">
<option value="mon">Monday</option>
<option value="tue">Tuesday</option>
<option value="wed">Wednesday</option>
</select>
</fieldset>
</form>
```

提示：如果您有一个带有相关选项的很长的列表，请在 <select> 内使用 <optgroup> 元素：

实例

```
<select name="day" id="day">
<optgroup label="Weekdays">
<option value="mon">Monday</option>
<option value="tue">Tuesday</option>
<option value="wed">Wednesday</option>
</optgroup>
<optgroup label="Weekends">
<option value="sat">Saturday</option>
<option value="sun">Sunday</option>
</optgroup>
</select>
```

自定义选择菜单

本页顶部的图像，演示了移动平台上如何使用它们的方式展示一个选择菜单。

如果您想要让选择菜单在所有的移动设备上都显示相同，请使用 jQuery 自带的自定义选择菜单，data-native-menu="false" 属性：

实例

```
<select name="day" id="day" data-native-menu="false">
```

多个选择

如需在选择菜单中选择多个选项，请在 <select> 元素中使用 multiple 属性：

实例

```
<select name="day" id="day" multiple data-native-menu="false">
```

更多实例

使用 data-role="controlgroup" 如何组合一个或多个选择菜单。

使用 data-type="horizontal" 如何水平组合选择菜单。

预选中选项 如何预选中一个选项。

可折叠表单 如何创建可折叠表单。

jQuery Mobile 表单滑动条

jQuery Mobile 滑动条控件

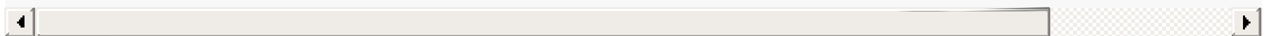
滑动条允许您从一个范围的数字中选择一个值：



如需创建滑动条，请使用 <input type="range">：

实例

```
<form method="post" action="demoform.html">
<div data-role="fieldcontain">
<label for="points">Points:</label>
<input type="range" name="points" id="points" value="50" min="0" ma
</div>
</form>
```



使用以下属性来规定限制：

- max - 规定允许的最大值
- min - 规定允许的最小值
- step - 规定合法的数字间隔
- value - 规定默认值

提示：如果您想要高亮突出显示滑动条的值，请添加 data-highlight="true"：

实例

```
<input type="range" data-hightlight="true">
```

拨动开关

波动开关通常用于 on/off 或 true/false 按钮：

如需创建一个开关，请把 <select> 元素与 data-role="slider" 一起使用，并添加两个 <option> 元素：

实例

```
<form method="post" action="demoform.html">
<div data-role="fieldcontain">
<label for="switch">Toggle Switch:</label>
<select name="switch" id="switch" data-role="slider">
<option value="on">On</option>
<option value="off">Off</option>
</select>
</div>
</form>
```

提示：请使用 "selected" 属性来设置选项中的一个为预选中状态（高亮突出显示状态）：

实例

```
<option value="off" selected>Off</option>
```

jQuery Mobile 主题

jQuery Mobile 主题

jQuery Mobile 提供了5种不同的主题样式, 从 "a" 到 "e" - 每一种主题的按钮，工具条，内容块等等颜色都不一致，每个主题的视觉效果也不一样。

通过设置元素的data-theme属性可以自定义应用的外观:

```
<div data-role="page" data-theme="a|b|c|d|e">
```

值	描述	实例
a	默认。黑色背景白色文字	
b	蓝色背景白色文字/ 黑色文字灰色背景	
c	黑色文字浅灰色背景	
d	黑色为主白色背景	
e	黑色文字橙色背景	

> 你喜欢混合主题！
>
> 默认情况下， jQuery Mobile 使用 "a" 主题 (黑色) 来设置头部和底部， "c" 主题 (浅灰色) 设置页面内容。 但是，你可以自定义设置你喜欢的混合主题。

主题头部，内容和底部

实例

```
<div data-role="header" data-theme="b"></div>

<div data-role="content" data-theme="a"></div>

<div data-role="footer" data-theme="e"></div>
```

主题对话框

实例

```
<a href="#pagetwo" data-rel="dialog">Go To The Themed Dialog Page</

<div data-role="page" id="pagetwo" data-overlay-theme="e">
  <div data-role="header" data-theme="b"></div>
  <div data-role="content" data-theme="a"></div>
  <div data-role="footer" data-theme="c"></div>
</div>
```

主题按钮

实例

```
<a href="#" data-role="button" data-theme="a">Button</a>
<a href="#" data-role="button" data-theme="b">Button</a>
<a href="#" data-role="button" data-theme="c">Button</a>
```

主题图标

实例

```
<a href="#" data-role="button" data-icon="plus" data-theme="e">Plus
```

头部和底部的主题按钮

实例

```
<div data-role="header">
  <a href="#" data-role="button" data-icon="home" data-theme="b">Ho
  <h1>Welcome To My Homepage</h1>
  <a href="#" data-role="button" data-icon="search" data-theme="e">
</div>

<div data-role="footer">
  <a href="#" data-role="button" data-theme="b" data-icon="plus">Bu
  <a href="#" data-role="button" data-theme="c" data-icon="plus">Bu
  <a href="#" data-role="button" data-theme="e" data-icon="plus">Bu
</div>
```

主题导航条

实例

```
<div data-role="footer" data-theme="e">
  <h1>Insert Footer Text Here</h1>
  <div data-role="navbar">
    <ul>
      <li><a href="#" data-icon="home" data-theme="b">Button 1</a><
      <li><a href="#" data-icon="arrow-r">Button 2</a></li>
      <li><a href="#" data-icon="arrow-r">Button 3</a></li>
      <li><a href="#" data-icon="search" data-theme="a" >Button 4</
    </ul>
  </div>
</div>
```

主题可折叠按钮和内容

实例

```
<div data-role="collapsible" data-theme="b" data-content-theme="e">
  <h1>Click me - I'm collapsible!</h1>
  <p>I'm the expanded content.</p>
</div>
```

主题列表

实例

```
<ul data-role="listview" data-theme="e">
  <li><a href="#">List Item</a></li>
  <li data-theme="a"><a href="#">List Item</a></li>
  <li data-theme="b"><a href="#">List Item</a></li>
  <li><a href="#">List Item</a></li>
</ul>
```

主题分割按钮

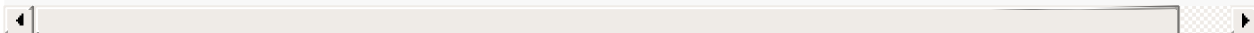
实例

```
<ul data-role="listview" data-split-theme="e">
```

主题可折叠列表

实例

```
<div data-role="collapsible" data-theme="b" data-content-theme="e">
  <ul data-role="listview">
    <li><a href="#">Agnes</a></li>
  </ul>
</div>
```



主题表单

实例

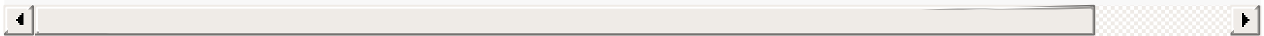
```
<label for="name">Full Name:</label>
<input type="text" name="text" id="name" data-theme="a">

<label for="colors">Choose Favorite Color:</label>
<select id="colors" name="colors" data-theme="b">
  <option value="red">Red</option>
  <option value="green">Green</option>
  <option value="blue">Blue</option>
</select>
```

主题可折叠表单

实例

```
<fieldset data-role="collapsible" data-theme="b" data-content-theme
<legend>Click me - I'm collapsible!</legend>
```



添加新主题

jQuery Mobile 可以在移动页面添加新主题。

通过修改 CSS 文件来添加或编辑新主题(如果你已经下载了 jQuery Mobile)。你只需要拷贝样式模块，然后重命令字母类名（f-z）,并在样式中添加你喜欢的颜色和字体。

你也可以在 HTML 文档中添加主题的新样式 - 工具条添加类： ui-bar-(a-z)， 文本内容添加类: ui-body-(a-z) for the content:

实例

```
<style>
.ui-bar-f
{
color:green;
background-color:yellow;
}
.ui-body-f
{
font-weight:bold;
color:purple;
}
</style>
```

jQuery Mobile 事件

事件 = 所有不同访问者访问页面的响应动作。

jQuery Mobile 事件

在jQuery Mobile你可以使用任何标准的 jQuery 事件 。

除此之外, jQuery Mobile 也提供了针对移动端浏览器的事件：

- 触摸事件 - 当用户触摸屏幕时触发
- 滑动事件 - 当用户上下滑动时触发
- 定位事件 - 当设备水平或垂直翻转时触发
- 页面事件 - 当页面显示，隐藏，创建，加载或未加载时触发

初始化 jQuery Mobile 事件

在学习jQuery时我们学到了用$(document).ready()来使你的jQuery代码脚本在DOM元素加载完成后才开始执行：

jQuery document ready 事件

```
<script>
$(document).ready(function(){

   _// jQuery methods go here...
_
});
</script>
```

但是，在 jQuery Mobile 中, 使用pageinit 事件来设置代码脚本在DOM元素加载完成后开始执行，所以要在任何新页面加载并创建是执行脚本，就需要绑定pageinit事件。

第二个参数 ("*#pageone*")为指定事件的页面id：

jQuery Mobile pageinit 事件

```
<script>
$(document).on("pageinit","#pageone",function(){

    _// jQuery events go here..._

});
</script>
```

>  注意： jQuery on() 方法用于绑定事件到选中的元素上。

下一章节我们将更详细介绍 jQuery Mobile 事件。

jQuery Mobile 触摸事件

触摸事件在用户触摸屏幕（页面）时触发。

点击或者滑动本区域


触摸事件同样可应用与桌面电脑上：点击或者滑动鼠标！

jQuery Mobile 点击

点击事件在用户点击元素时触发。

如下实例：当点击 <p> 元素时，隐藏当前的 <p> 元素：

实例

```
$("p").on("tap",function(){
  $(this).hide();
});
```

jQuery Mobile 点击不放（长按）

点击不放（长按） 事件在点击并不放（大约一秒）后触发

实例

```
$("p").on("taphold",function(){
  $(this).hide();
});
```

jQuery Mobile 滑动

滑动事件是在用户一秒内水平拖拽大于30PX，或者纵向拖曳小于20px的事件发生时触发的事件：

实例

```
$("p").on("swipe",function(){
  $("span").text("Swipe detected!");
});
```

jQuery Mobile 向左滑动

向左滑动事件在用户向左拖动元素大于30px时触发：

实例

```
$("p").on("swipeleft",function(){
  alert("You swiped left!");
});
```

jQuery Mobile 向右滑动

向右滑动事件在用户向右拖动元素大于30px时触发：

实例

```
$("p").on("swiperight",function(){
  alert("You swiped right!");
});
```

jQuery Mobile 滚屏事件

jQuery Mobile 提供了两种滚屏事件：滚屏开始时触发和滚动结束时触发。

jQuery Mobile 滚屏开始（Scrollstart）

scrollstart 事件是在用户开始滚动页面时触发：

实例

```
$(document).on("scrollstart",function(){
alert("Started scrolling!");
});
```

>  注意：iOS 设备在滚屏时锁定 DOM 操作，这意味着当用户滚屏时不可能改变任何东西。然而，jQuery 团队正在为此寻找解决方案。

jQuery Mobile 滚屏结束（Scrollstop）

scrollstop 事件是在用户停止滚动页面时触发：

实例

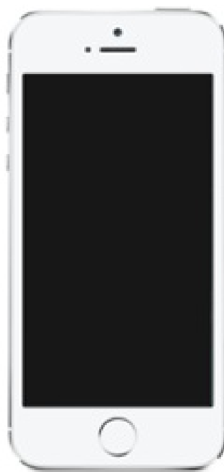
```
$(document).on("scrollstop",function(){
alert("Stopped scrolling!");
});
```

jQuery Mobile 方向改变事件

jQuery Mobile 方向改变（orientationchange）事件

当用户垂直或水平旋转移动设备时，触发方向改变（orientationchange）事件。

水平旋转　　　　垂直旋转



如需使用方向改变（orientationchange）事件，请附加它到 window 对象：

```
$(window).on("orientationchange",function(){
alert("The orientation has changed!");
});
```

回调函数可有一个参数，event 对象，返回移动设备的方向："纵向"（设备保持在垂直位置）或"横向"（设备保持在水平位置）：

实例

```
$(window).on("orientationchange",function(event){
alert("Orientation is: " + event.orientation);
});
```

由于方向改变（orientationchange）事件绑定到 window 对象，我们可以使用 window.orientation 属性来设置不同的样式，以便区分纵向和横向的视图：

实例

```
$(window).on("orientationchange",function(){
if(window.orientation == 0) // Portrait
{
$("p").css({"background-color":"yellow","font-size":"300%"});
}
else // Landscape
{
$("p").css({"background-color":"pink","font-size":"200%"});
}
});
```

>  window.orientation 属性针对纵向视图返回 0，针对横向视图返回 90 或 -90。

jQuery Mobile Data 属性

jQuery Data 属性

jQuery Mobile 使用 HTML5 data-* 属性来为移动设备创建更具触摸友好性和吸引性的外观。

在下面的参考列表中，粗体显示的值为默认值。

按钮

带有 data-role="button" 的超链接。button 元素、工具栏中的链接以及 input 字段都会自动渲染成按钮样式，不需要添加 data-role="button"。

Data-属性	值	描述
data-corners	**true** | false	规定按钮是否圆角
data-icon	Icons 参考手册	规定按钮的图表。默认没有图标。
data-iconpos	**left** | right | top | bottom | notext	规定图标的位置
data-iconshadow	**true** | false	规定按钮图标是否有阴影
data-inline	true | **false**	规定按钮是否内联
data-mini	true | **false**	规定按钮是小尺寸还是常规尺寸
data-shadow	**true** | false	规定按钮是否有阴影
data-theme	*letter* (a-z)	规定按钮的主题颜色


如需组合多个按钮，请使用带有 data-role="controlgroup" 属性和 data-type="horizontal|vertical" 的容器来规定是否水平或垂直组合按钮。

复选框

带有 type="checkbox" 的成双成对的 label 和 input。它们会被自动渲染程按钮样式，data-role 不是必需的。

Data-属性	值	描述
data-mini	true | **false**	规定复选框是小尺寸还是常规尺寸
data-role	none	防止 jQuery Mobile 把复选框渲染成按钮样式
data-theme	*letter* (a-z)	规定复选框的主题颜色

 如需组合多个复选框，请使用带有 data-role="controlgroup" 属性和 data-type="horizontal|vertical" 的容器来规定是否水平或垂直组合复选框。

可折叠块

在带有 data-role="collapsible" 的容器内部的后跟任意 HTML 标记的标题元素。

Data-属性	值	描述
data-collapsed	**true** | false	规定内容是折叠还是展开
data-collapsed-icon	Icons 参考手册	规定可折叠按钮的图标。默认是 "plus"
data-content-theme	*letter* (a-z)	规定可折叠内容的主题颜色。是否为可折叠内容添加圆角
data-expanded-icon	Icons 参考手册	规定当内容展开时可折叠按钮的图标。默认是 "minus"
data-iconpos	**left** | right | top | bottom	规定图标的位置
data-inset	**true** | false	规定可折叠按钮是否渲染成圆角且带外边距
data-mini	true | **false**	规定可折叠按钮是小尺寸还是常规尺寸
data-theme	*letter* (a-z)	规定可折叠按钮的主题颜色

可折叠集合

在带有 data-role="collapsible-set" 的容器内部的可折叠内容块。

Data-属性	值	描述
data-collapsed-icon	Icons 参考手册	规定可折叠按钮的图标。默认是 "plus"
data-content-theme	*letter* (a-z)	规定可折叠按钮的主题颜色
data-expanded-icon	Icons 参考手册	规定当内容展开时可折叠按钮的图标。默认是 "minus"
data-iconpos	**left** | right | top | bottom | notext	规定图标的位置
data-inset	**true** | false	规定可折叠块是否渲染成圆角且带外边距
data-mini	true | **false**	规定可折叠按钮是小尺寸还是常规尺寸
data-theme	*letter* (a-z)	规定可折叠集合的主题颜色

内容

带有 data-role="content" 的容器。

Data-属性	值	描述
data-theme	*letter* (a-z)	规定内容的主题颜色。默认是 "c"

控件组

带有 data-role="controlgroup" 的 <div> 或 <fieldset> 容器。 组合单个类型（基于链接的按钮、单选按钮、复选框、select 元素）的多个按钮样式的 input。对于组合表单复选框和单选按钮，推荐在带有 data-role="fieldcontain" 的 <div> 内使用 <fieldset> 容器来改进标签样式。

Data-属性	值	描述
data-mini	true | **false**	规定控件组是小尺寸还是常规尺寸
data-type	horizontal | **vertical**	规定控件组是水平显示还是垂直显示

对话框

带有 data-role="dialog" 的容器或带有 data-rel="dialog" 的链接。

Data-属性	值	描述
data-close-btn-text	*sometext*	规定对话框关闭按钮的文本
data-dom-cache	true | **false**	规定是否清除各个页面的 jQuery DOM 缓存（如果设置为 true，您必须自己管理 DOM 并在所有移动设备上进行测试）
data-overlay-theme	*letter* (a-z)	规定对话框页面的蒙版（背景）颜色
data-theme	*letter* (a-z)	规定对话框页面的主题颜色
data-title	*sometext*	规定对话框页面的标题

增强

带有 data-enhance="false" 或 data-ajax="false" 的容器。

Data-属性	值	描述
data-enhance	**true** | false	如果设置为 "true"（默认），jQuery Mobile 会自动渲染页面，使其更适合于移动设备。如果设置为 "false"，框架将不会渲染页面
data-ajax	**true** | false	规定是否通过 ajax 加载页面

注意：data-enhance="false" 必须与 $.mobile.ignoreContentEnabled=true" 一同使用来阻止 jQuery Mobile 自动渲染页面。

当 $.mobile.ignoreContentEnabled 设置为 true 时，data-ajax="false" 容器内的任何链接或表单元素将会被框架的导航功能忽略。

域容器

包围在 label/表单元素对周围的带有 data-role="fieldcontain" 的容器。

固定工具栏

带有 data-role="header" 或 data-role="footer"，并带有 data-position="fixed" 属性的容器。

Data-属性	值	描述
data-disable-page-zoom	**true** | false	规定用户是否能缩放页面
data-fullscreen	true | **false**	规定工具栏是否一直固定在顶部或底部
data-tap-toggle	**true** | false	规定用户是否能够通过点击改变工具栏的可见性
data-transition	**slide** | fade | none	规定当点击发生时的切换效果
data-update-page-padding	**true** | false	规定页面顶部和底部的内边距是否在 resize、transition 和 "updatelayout" 事件发生时更新（jQuery Mobile 在 "pageshow" 事件发生时总是更新内边距）
data-visible-on-page-show	**true** | false	规定当父页面显示时工具栏的可见性

翻转拨动开关

一个带有 data-role="slider" 的 <select> 元素和两个 <option> 元素。

Data-属性	值	描述
data-mini	true | **false**	规定开关是小尺寸还是常规尺寸
data-role	none	防止 jQuery Mobile 把拨动开关渲染成按钮样式
data-theme	*letter* (a-z)	规定拨动开关的主题颜色
data-track-theme	*letter* (a-z)	规定轨道的主题颜色

尾部栏

带有 data-role="footer" 的容器。

Data-属性	值	描述
data-id	*sometext*	规定唯一 ID。对于持续的尾部栏是必需的
data-position	**inline** | fixed	规定尾部栏是与页面内容内联还是保持固定在底部
data-fullscreen	true | **false**	规定尾部栏是固定在底部还是覆盖在页面内容上（查看范围更大）
data-theme	*letter* (a-z)	规定尾部栏的主题颜色。默认是 "a"

注意：如需启用全屏定位，请使用 data-position="fixed"，然后添加 data-fullscreen 属性到元素上。

头部栏

带有 data-role="header" 的容器。

Data-属性	值	描述
data-id	*sometext*	规定唯一 ID。对于持续的头部栏是必需的
data-position	**inline** | fixed	规定头部栏是与页面内容内联还是保持固定在顶部
data-fullscreen	true | **false**	规定头部栏是固定在顶部还是覆盖在页面内容上（查看范围更大）
data-theme	*letter* (a-z)	规定头部栏的主题颜色。默认是 "a"

注意：如需启用全屏定位，请使用 data-position="fixed"，然后添加 data-fullscreen 属性到元素上。

链接

所有的链接，包含那些带有 data-role="button" 的链接和表单提交按钮。

Data-属性	值	描述
data-ajax	**true** | false	规定是否通过 ajax 加载页面来提高用户体验和交互。如果设置为 false，jQuery Mobile 将会执行一个正常的页面请求。
data-direction	reverse	反向转换动画（仅用于页面和对话框）
data-dom-cache	true | **false**	规定是否清除各个页面的 jQuery DOM 缓存（如果设置为 true，您必须自己管理 DOM 并在所有移动设备上进行测试）
data-prefetch	true | **false**	规定是否预先读取页面到 DOM 中，以便当用户访问它们时可用
data-rel	back | dialog | external | popup	规定链接行为的选项。Back - 回退到历史记录中的前一个页面。Dialog - 以对话框形式打开链接，不保存到历史记录中。External - 用于链接到另一个域。Popup - 打开一个弹出窗口。
data-transition	**fade** | flip | flow | pop | slide | slidedown | slidefade | slideup | turn | none	规定一个页面切换到下一个页面的效果。请查看我们的 jQuery Mobile 页面切换章节。
data-position-to	**origin** | jQuery selector | window	规定弹出框的位置。Origin - 默认。定位弹窗在打开它的链接上。jQuery selector - 定位弹窗在指定元素上。Window - 定位弹窗在窗口屏幕的中央。

列表

带有 data-role="listview" 的 <ol> 或 。

Data-属性	值	描述
data-autodividers	true | **false**	规定是否自动划分列表项
data-count-theme	*letter* (a-z)	规定计数气泡的主题颜色。默认是 "c"
data-divider-theme	*letter* (a-z)	规定列表分隔的主题颜色。默认是 "b"
data-filter	true | **false**	规定是否在列表中添加搜索框
data-filter-placeholder	*sometext*	规定搜索框内的文本。默认是 "Filter items..."
data-filter-theme	*letter* (a-z)	规定搜索过滤的主题颜色。默认是 "c"
data-icon	Icons 参考手册	规定列表的图标
data-inset	true | **false**	规定列表是否渲染成圆角且带外边距
data-split-icon	Icons 参考手册	规定分割按钮的图表。默认是 "arrow-r"
data-split-theme	*letter* (a-z)	规定分割按钮的主题颜色。默认是 "b"
data-theme	*letter* (a-z)	规定列表的主题颜色

列表项

带有 data-role="listview" 的 <ol> 或 <ul> 内的 。

Data-属性	值	描述
data-filtertext	*sometext*	规定当过滤元素时要搜索的文本。该文本将会被过滤，而不是实际的列表项文本
data-icon	Icons 参考手册	规定列表项图标
data-role	list-divider	规定列表项的分隔
data-theme	*letter* (a-z)	规定列表项的主题颜色

注意：data-icon 属性只作用于带有链接的列表项。

导航栏

带有 data-role="navbar" 容器内部的 <li> 元素。

Data-属性	值	描述
data-icon	Icons 参考手册	规定列表项的图标
data-iconpos	left | right | **top** | bottom | notext	规定图标的位置

 导航栏从他们的父容器中继承了主题样本。为导航栏设置 data-theme 属性是不可能的。可以为导航栏中的每个链接单独设置 data-theme 属性。

页面

带有 data-role="page" 的容器。

Data-属性	值	描述
data-add-back-btn	true | **false**	自动添加后退按钮，仅限头部栏
data-back-btn-text	*sometext*	规定后退按钮的一些文本
data-back-btn-theme	*letter* (a-z)	规定后退按钮的主题颜色
data-close-btn-text	*letter* (a-z)	规定对话框上关闭按钮的一些文本
data-dom-cache	true | **false**	规定是否清除各个页面的 jQuery DOM 缓存（如果设置为 true，您必须自己管理 DOM 并在所有移动设备上进行测试）
data-overlay-theme	*letter* (a-z)	规定对话框页面的蒙版（背景）颜色
data-theme	*letter* (a-z)	规定页面的主题颜色。默认是 "c"
data-title	*sometext*	规定页面标题
data-url	url	更新 URL 的值，而不是用于请求页面的 URL

弹窗

带有 data-role="popup" 的容器。

Data-属性	值	描述
data-corners	**true** | false	规定弹窗是否圆角
data-overlay-theme	*letter* (a-z)	规定弹出框的蒙版（背景）颜色。默认是透明背景（无）
data-shadow	**true** | false	规定弹出框是否有阴影
data-theme	*letter* (a-z)	规定弹出框的主题颜色。默认是继承的，"none" 设置弹窗为透明的
data-tolerance	30, 15, 30, 15	规定窗口边缘（上 top、右 right、下 bottom、左 left）的距离

带有 data-rel="popup" 的锚：

Data-属性	值	描述
data-position-to	**origin** | jQuery selector | window	>规定弹出框的位置。Origin - 默认。定位弹窗在打开它的链接上。jQuery selector - 定位弹窗在指定元素上。Window - 定位弹窗在窗口屏幕的中央。
data-rel	popup	用于打开弹出框
data-transition	fade | flip | flow | pop | slide | slidedown | slidefade | slideup | turn | **none**	规定一个页面切换到下一个页面的效果。请查看我们的 jQuery Mobile 页面切换章节。

单选按钮

带有 type="radio" 的成双成对的 label 和 input。它们会被自动渲染程按钮样式，data-role 不是必需的。

Data-属性	值	描述
data-mini	true | **false**	规定按钮是小尺寸还是常规尺寸
data-role	none	防止 jQuery Mobile 渲染单选按钮的样式为增强状态的按钮，使元素以 HTML 原生的状态显示
data-theme	*letter* (a-z)	规定单选按钮的主题颜色

 如需组合多个单选按钮，请使用带有 data-role="controlgroup" 属性和 data-type="horizontal|vertical" 的容器来规定是否水平或垂直组合单选按钮。

选择

所有的 <select> 元素。这些会被自动渲染成按钮样式，date-role 是不必需的。

Data-属性	值	描述
data-icon	Icons 参考手册	规定 select 元素的图标。默认是 "arrow-d"
data-iconpos	left | **right** | top | bottom | notext	规定图标的位置
data-inline	true | **false**	规定 select 元素是否内联
data-mini	true | **false**	规定 select 元素是小尺寸还是常规尺寸
data-native-menu	**true** | false	当设置为 false 时，它使用 jQuery 自带的自定义的选择菜单（如果您想要让选择菜单在所有的移动设备上都显示相同，则推荐这么使用）
data-overlay-theme	*letter* (a-z)	规定 jQuery 自带的自定义的选择菜单的主题颜色（与 data-native-menu="false" 一起使用）
data-placeholder	true | **false**	可在一个非原生的选择菜单的 <option> 元素上设置
data-role	none	防止 jQuery Mobile 把 select 元素渲染成按钮样式
data-theme	*letter* (a-z)	规定 select 元素的主题颜色

>  如需组合多个 select 元素，请使用带有 data-role="controlgroup" 属性和 data-type="horizontal|vertical" 的容器来规定是否水平或垂直组合 select 元素。

滑动块

带有 type="range" 的输入框。这些会被自动渲染成按钮样式，date-role 是不必需的。

Data-属性	值	描述
data-highlight	true | **false**	规定滑动轨道是否高亮突出显示
data-mini	true | **false**	规定滑动块是小尺寸还是常规尺寸
data-role	none	防止 jQuery Mobile 渲染滑动块控件为按钮样式
data-theme	*letter* (a-z)	规定滑动块控件（输入框、手柄和轨道）的主题颜色
data-track-theme	*letter* (a-z)	规定滑动块轨道的主题颜色

文本输入框 & 文本输入域

带有 type="text|search|etc." 的 input 或 textarea 元素。这些会被自动渲染成按钮样式，date-role 是不必需的。

Data-属性	值	描述
data-mini	true | **false**	规定输入框是小尺寸还是常规尺寸
data-role	none	防止 jQuery Mobile 把输入框/输入域渲染成按钮样式
data-theme	*letter* (a-z)	规定输入字段的主题颜色

jQuery Mobile 图标

jQuery 图标

在 jQuery Mobile 中，如需为按钮添加图标，请使用 data-icon 属性：

```
<a href="#anylink" data-role="button" **data-icon="refresh"**>Refre
```

提示：在 <button> 或 <input> 元素中，您也可以使用 data-icon 属性。

下面我们列出了 jQuery Mobile 提供的所有可用图标：

属性值	描述	图标	实例
data-icon="arrow-l"	左箭头		
data-icon="arrow-r"	右箭头		
data-icon="arrow-u"	上箭头		
data-icon="arrow-d"	下箭头		
data-icon="plus"	加号		
data-icon="minus"	减号		
data-icon="delete"	删除		
data-icon="check"	检查		
data-icon="home"	首页		
data-icon="info"	信息		
data-icon="grid"	网格		
data-icon="gear"	工具		
data-icon="search"	搜索		
data-icon="back"	后退		
data-icon="forward"	前进		
data-icon="refresh"	更新		
data-icon="star"	星号		
data-icon="alert"	警告		

jQuery Mobile 事件

jQuery Mobile 事件参考手册

以下列表为所有的 jQuery Mobile 事件。

注意：请使用 on() 方法绑定事件。

事件	描述
hashchange	启用可标记 #hash 历史，哈希值会在一次独立的点击时发生时变化，比如一个用户点击后退按钮，会通过hashchange事件进行处理。
navigate	包裹了 hashchange 和 popstate 的事件
orientationchange	方向改变事件,在用户垂直或者水平旋转移动设备时触发。
pagebeforechange	在页面切换之前，触发的事件。使用$.mobile.changePage()来切换页面，此方法触发2个事件，切换之前的pagebeforechange事件，和切换完成后pagechange（成功）或者pagechangefailed（失败）。
pagebeforecreate	页面初始化时，初始化之前触发。
pagebeforehide	在页面切换后旧页面隐藏之前，触发的事件。
pagebeforeload	在加载请求发出之前触发
pagebeforeshow	在页面切换后显示之前，触发的事件。
pagechange	在页面切换成功后，触发的事件。使用$.mobile.changePage()来切换页面，此方法触发2个事件，切换之前的pagebeforechange事件，和切换完成后pagechange（成功）或者pagechangefailed（失败）。
pagechangefailed	在页面切换失败时，触发的事件。使用$.mobile.changePage()来切换页面，此方法触发2个事件，切换之前的pagebeforechange事件，和切换完成后pagechange（成功）或者pagechangefailed（失败）。
pagecreate	在页面创建成功之后，触发的事件,但增强完成之前。
pagehide	在页面切换后老页面隐藏之后，触发的事件。
pageinit	在页面页面初始化时，触发的事件。
pageload	在页面完全加载成功后触发。

pageloadfailed	如果页面请求失败触发。
pageremove	在窗口视图从 DOM 中移除外部页面之前触发。
pageshow	在过渡动画完成后，在"到达"页面触发。
scrollstart	当用户开始滚动页面时触发。
scrollstop	当用户停止滚动页面时触发。
swipe	当用户在元素上水平滑动时触发。
swipeleft	当用户从左划过元素超过 30px 时触发。
swiperight	当用户从右划过元素超过 30px 时触发。
tap	当用户敲击某元素时触发。
taphold	当元素敲击某元素并保持一秒时触发。
throttledresize	启用可标记 #hash 历史记录
updatelayout	由动态显示/隐藏内容的 jQuery Mobile 组件触发。
vclick	虚拟化的 click 事件处理器
vmousecancel	虚拟化的 mousecancel 事件处理器
vmousedown	虚拟化的 mousedown 事件处理器
vmousemove	虚拟化的 mousemove 事件处理器
vmouseout	虚拟化的 mouseout 事件处理器
vmouseover	虚拟化的 mouseover 事件处理器
vmouseup	虚拟化的 mouseup 事件处理器

jQuery Mobile 页面事件

jQuery Mobile 页面事件

在 jQuery Mobile 中与页面打交道的事件被分为四类：

- Page Initialization - 在页面创建前，当页面创建时，以及在页面初始化之后
- Page Load/Unload - 当外部页面加载时、卸载时或遭遇失败时
- Page Transition - 在页面过渡之前和之后
- Page Change - 当页面被更改，或遭遇失败时

如需关于所有 jQuery Mobile 事件的完整信息，请访问我们的 jQuery Mobile 事件参考手册。

jQuery Mobile Initialization 事件

当 jQuery Mobile 中的一张典型页面进行初始化时，它会经历三个阶段：

- 在页面创建前
- 页面创建
- 页面初始化

每个阶段触发的事件都可用于插入或操作代码。

事件	描述
pagebeforecreate	当页面即将初始化，并且在 jQuery Mobile 已开始增强页面之前，触发该事件。
pagecreate	当页面已创建，但增强完成之前，触发该事件。
pageinit	当页面已初始化，并且在 jQuery Mobile 已完成页面增强之后，触发该事件。

下面的例子演示在 jQuery Mobile 中创建页面时，何时触发每种事件：

实例

```
$(document).on("pagebeforecreate",function(event){
  alert("触发 pagebeforecreate事件！");
});
$(document).on("pagecreate",function(event){
  alert("触发 pagecreate 事件！");
});
$(document).on("pageinit",function(event){
  alert("触发 pageinit 事件！")
});
```

jQuery Mobile Load 事件

页面加载事件属于外部页面。

无论外部页面何时载入 DOM，将触发两个事件。第一个是 pagebeforeload，第二个是 pageload （成功）或 pageloadfailed（失败）。

下表中解释了这些事件：

事件	描述
pagebeforeload	在任何页面加载请求作出之前触发。
pageload	在页面已成功加载并插入 DOM 后触发。
pageloadfailed	如果页面加载请求失败，则触发该事件。默认地，将显示 "Error Loading Page" 消息。

下列演示 pageload 和 pagloadfailed 事件的工作原理：

实例

```
$(document).on("pageload",function(event,data){
  alert("触发 pageload 事件！\nURL: " + data.url);
});
$(document).on("pageloadfailed",function(event,data){
  alert(";抱歉，被请求页面不存在。");
});
```

jQuery Mobile 过渡事件

我们还可以在从一页过渡到下一页时使用事件。

页面过渡涉及两个页面：一张"来"的页面和一张"去"的页面 - 这些过渡使当前活动页面（"来的"页面）到新页面（"去的"页面的改变过程变得更加动感。

事件	描述
pagebeforeshow	在"去的"页面触发，在过渡动画开始前。
pageshow	在"去的"页面触发，在过渡动画完成后。
pagebeforehide	在"来的"页面触发，在过渡动画开始前。
pagehide	在"来的"页面触发，在过渡动画完成后。

下列演示了过渡时间的工作原理：

实例

```
$(document).on("pagebeforeshow","#pagetwo",function(){ //当进入页面二
  alert("页面二即将显示");
});
$(document).on("pageshow","#pagetwo",function(){ // 当进入页面二时
  alert("现在显示页面二");
});
$(document).on("pagebeforehide","#pagetwo",function(){ // 当进入页面
  alert("页面二即将隐藏");
});
$(document).on("pagehide","#pagetwo",function(){ // When leaving pa
  alert("现在隐藏页面二");
});
```

jQuery Mobile CSS 类

jQuery CSS 类

jQuery Mobile CSS 类来设置不同元素的样式。

以下列表包含了通用的 CSS 样式：

全局类

以下类可以在 jQuery Mobile 小工具中使用 (按钮，工具条，面板，表格，列表等。):

Class	描述
ui-corner-all	为元素添加圆角
ui-shadow	为元素添加阴影
ui-overlay-shadow	为元素添加多层阴影
ui-mini	让元素变小

按钮类

除了全局类外，你可以向 <a> 或 <button> 元素添加以下类 (不是 <input> 按钮):

Class	描述
ui-btn	添加在 <a> 元素上并以按钮形式展示
ui-btn-inline	在同一行上显示按钮
ui-btn-icon-top	定位图标在按钮文本之上
ui-btn-icon-right	定位图标在按钮文本的右边
ui-btn-icon-bottom	定位图标在按钮文本之下
ui-btn-icon-left	定位图标在按钮文本的左边
ui-btn-icon-notext	只显示图标
ui-btn-a|b	指定按钮演示。"a" 是默认的 (灰色背景黑色文本样式), "b" 修改颜色为黑色背景白色文本

图标类

所有可用图片的类用在 <a> 和 <button> 元素上 (查看 jQuery Mobile 图标参考手册了解如何在其他元素上使用):

Class	图标描述	图标
ui-icon-action	动作 (一般用于页面跳转切换)	
ui-icon-alert	三角形内的感叹号	
ui-btn-audio	音响/音箱	
ui-icon-arrow-d-l	左下角箭头	
ui-icon-arrow-d-r	右下角箭头	
ui-icon-arrow-u-l	左上角箭头	
ui-icon-arrow-u-r	右上角箭头	
ui-icon-arrow-l	左角箭头	
ui-icon-arrow-r	右角箭头	

ui-icon-arrow-u	向上箭头	
ui-icon-arrow-d	向下箭头	
ui-icon-back	返回	
ui-icon-bars	三条水平线，一般用于展示列表按钮图标	
ui-icon-bullets	用于展示列表按钮图标	
ui-icon-calendar	日历	
ui-icon-camera	相机	
ui-icon-carat-d	向下滑动图标	
ui-icon-carat-l	向左滑动图标	
ui-icon-carat-r	向右滑动图标	
ui-icon-carat-u	向上滑动图标	
ui-icon-check	勾选	
ui-icon-clock	闹钟	
ui-icon-cloud	云	
ui-icon-comment	评论 / 消息	
ui-icon-delete	删除	
ui-icon-edit	编辑 / 铅笔	
ui-icon-eye	眼睛	
ui-icon-forbidden	禁用标识 sign	
ui-icon-forward	撤销 - (返回上一步)	
ui-icon-gear	齿轮，一般用于设置按钮图标	
ui-icon-grid	网格	
ui-icon-heart	心型，可用于文章收藏	

ui-icon-home	主页	
ui-icon-info	信息	
ui-icon-location	定位	
ui-icon-lock	锁	
ui-icon-mail	邮件 / 信封	
ui-icon-minus	减号	
ui-icon-navigation	导航	
ui-icon-phone	电话	
ui-icon-power	开关 (On/off)	
ui-icon-plus	加号	
ui-icon-recycle	循环 标识	
ui-icon-refresh	刷新	
ui-icon-search	搜索 / 放大镜	
ui-icon-shop	商店/购物袋	
ui-icon-star	星号	
ui-icon-tag	标签	
ui-icon-user	用户 / 人物	
ui-icon-video	视频 / 相机	

主题类 Classes

jQuery Mobile 提供了两个主题类: a (灰) 和 b (黑)。 但是，你可以创建你自己的自定义类 - 可定义到字母 "z" （查看 jQuery Mobile 主题 ） 。 下表列出了可用的主题类。 字母 (a-z) 意为样式可以指定 a 到 z。例如 ui-bar-a 或 ui-bar-b等。

Class	描述
ui-bar-(a-z)	指定栏目演示 - 头部，底部及其他栏目
ui-body-(a-z)	指定内容块的颜色 - 页面内容部分 (1.4.0 版本已废弃), 列表视图， 弹窗, 侧栏， 面板，加载， 折叠。
ui-btn-(a-z)	指定按钮颜色
ui-group-theme-(a-z)	定义了控制组的演示 listviews 和 collapsible 集合。
ui-overlay-(a-z)	定义了页面背景颜色，包括对话框， 弹窗和其他出现在最顶层的页面容器
ui-page-theme-(a-z)	定义了页面演示

网格类

网格中的列是等宽的（合计是 100%）， 没有边框、背景、margin 或 padding。 这里有四种布局网格可供使用：

网格类	列	列宽	对应	实例
ui-grid-solo	1	100%	ui-block-a	
ui-grid-a	2	50% / 50%	ui-block-a|b	
ui-grid-b	3	33% / 33% / 33%	ui-block-a|b|c	
ui-grid-c	4	25% / 25% / 25% / 25%	ui-block-a|b|c|d	
ui-grid-d	5	20% / 20% / 20% / 20% / 20%	ui-block-a|b|c|d|e	

更多信息可以查看 jQuery Mobile 网格 章节。